Automotive Control Systems (MEEM/EE5812)

Model Predictive Control of a Power-Split HEV

Instructor

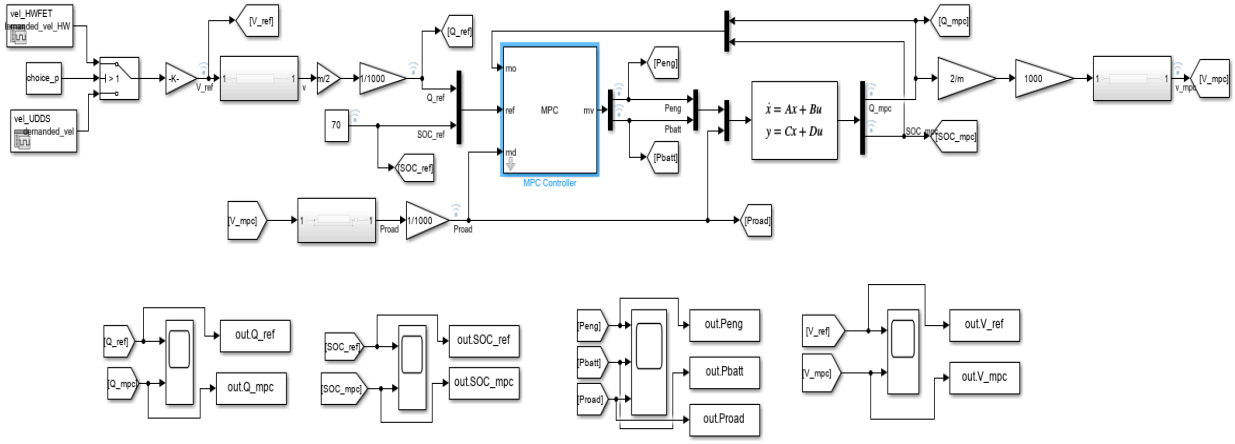
Dr. Bo Chen

Submitted By

Rahul Srikanth Kandi

Objective :

Our aim is to Design and implement a MPC (Model Predictive Controller) for a power split Hybrid electric vehicle for two drive cycles namely, UDDS (Urban dynamometer driving schedule) and HWFET (Highway fuel economy test). The controller should pass all the required power ramp, velocity and SOC limits and conditions to be a good one to use. We analyze the controller by changing individual weights for the Engine power, Battery power, Q (kinetic energy) and SOC variables and observe the impact on the system performance.



Above is the system model and the blue highlighted block is the MPC controller, we use it to optimize and analyze the system with the respective weights of input and output variables.

Parameter Name	Value
Maximum engine power	57kW
Maximum battery charging power	-23kW
Maximum battery discharging power	31kW
Minimum battery SOC	65%
Maximum battery SOC	75%

```
% A, B, C, D matrices [Peng, Pbatt, Proad]
A = [0 0; 0 0];
B = [1 1 -1; 0 -0.0182 0];
C = [1 0; 0 1];
D = zeros(2, 3);

plant_c = ss(A, B, C, D);
Ts = 0.1;
mpcPlant = c2d(plant_c, Ts);
```

We wrote the plant model in matlab by state space representation.

SOC & Power Dynamics.

The model captures how battery's SOC (state of charge) changes based on power drawn from it. Defined by a linear relation,

$$\dot{SOC} = a_1 \times P_{batt}$$

also $a_1 = -1.82 \times 10^{-7}$

This linear relation defines that the SOC decreases when battery supplies power and increases when it's being charged.

We also use power balance to observe how the vehicle's required power $\dot{Q}$ is split b/w engine, battery & power loss due to road load.

$$\dot{Q} = P_{eng} + P_{batt} - P_{road}.$$

The road load power P_{road} includes all resistive forces acting on the vehicle, such as,

→ Aerodynamic drag, → Rolling resistance, → road grade.

$$P_{road} = \left[\frac{1}{2}\rho C_d A(v + v_w)^2 + f_r mg\cos(\alpha) + mg\sin\alpha\right] \times v$$

STATE-SPACE MODEL :-

State vector :- $X = [Q \;\; SOC]^T$

Control Input vector :- $U = [P_{eng} \;\; P_{batt}]^T$

Output vector :- $Y = [Q \;\; SOC]^T$

Therefore,

State Space Equations :-

$$\dot{X} = AX + B_u U + B_w W$$

$$Y = CX.$$

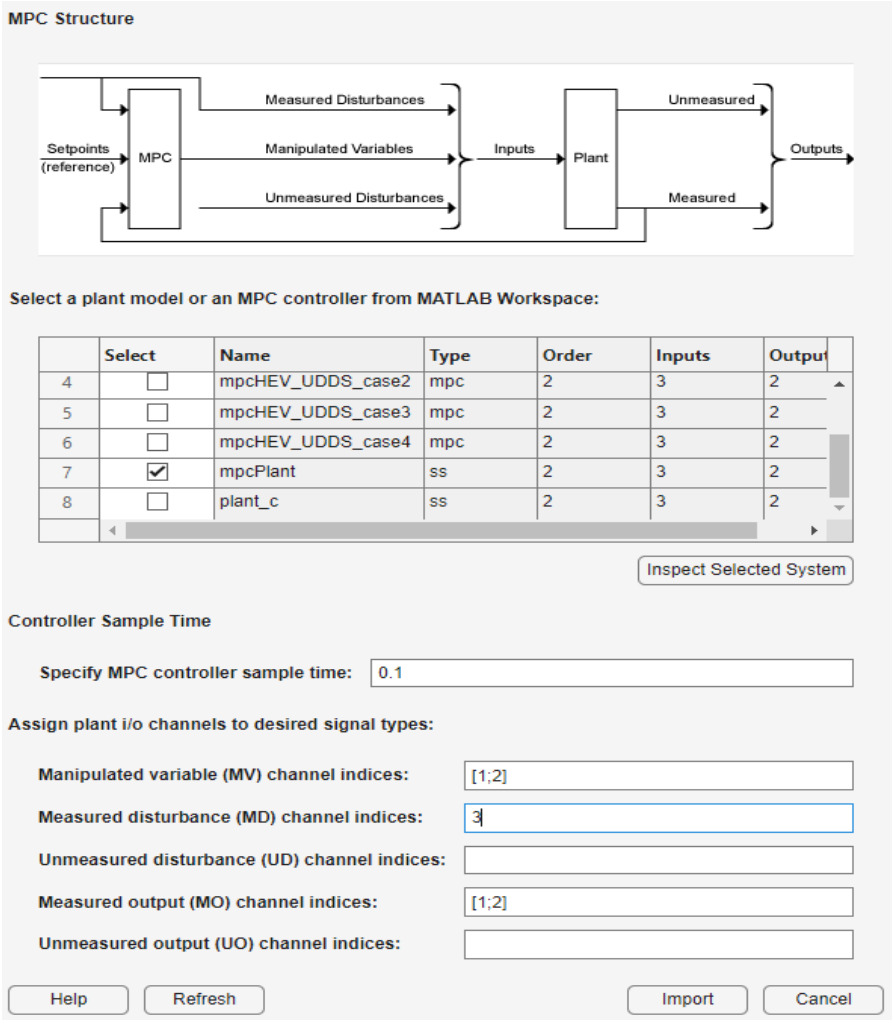
i.e

$$\dot{X} = AX + [B_u \;\; B_w]\begin{bmatrix} U \\ W \end{bmatrix}$$

$$Y = CX.$$

MPC STRUCTURE :

To design the MPC controller for the power-split HEV, I used a linear state-space model with three inputs—engine power (Peng), battery power (Pbatt), and road load disturbance (Proad)—and two outputs—engine torque (Q) and battery SOC. I set the controller's sample time to 0.1 seconds and chose a prediction and control horizon of 20 steps to balance accuracy and responsiveness. Constraints were applied to both the inputs and outputs to ensure realistic operation. In particular, SOC was strictly limited to stay between 65% and 75%, and smooth transitions were encouraged by assigning higher weights to the rate of change of control inputs. Output weights were set to emphasize SOC tracking more than torque tracking. This setup allowed the controller to make forward-looking decisions that respect both physical limits and performance goals.



Above is the start of Mpc structure modelling and where we use the linearized plant model with specifying the manipulated variables as Peng and Pbatt and measured disturbance as Proad. Along with this we are specifying two output indices.

Simulation Settings

Plant used in simulation: Default (controller internal model)

Simulation duration (seconds): 10

☐ Run open-loop simulation ☐ Use unconstrained MPC

☐ Preview references (look ahead) ☐ Preview measured disturbances (look ahead)

Reference Signals (setpoints for all outputs)

	Channel	Name	Nominal	Signal	Size	Time	Period
1	r(1)	Ref of MO1	0	Constant			
2	r(2)	Ref of MO2	0	Constant			

Measured Disturbances (inputs to MD channels)

	Channel	Name	Nominal	Signal	Size	Time	Period
1	u(3)	MD1	0	Step	1	1	

Output Disturbances (added at MO channels)

	Channel	Name	Nominal	Signal	Size	Time	Period
1	y(1)	MO1	0	Constant			
2	y(2)	MO2	0	Constant			

Load Disturbances (added at MV channels)

	Channel	Name	Nominal	Signal	Size	Time	Period
1	u(1)	MV1	0	Constant			
2	u(2)	MV2	0	Constant			

Above is the settings for signal change in the scenario of the MPC controller.

Input and Output Channel Specifications

Plant Inputs

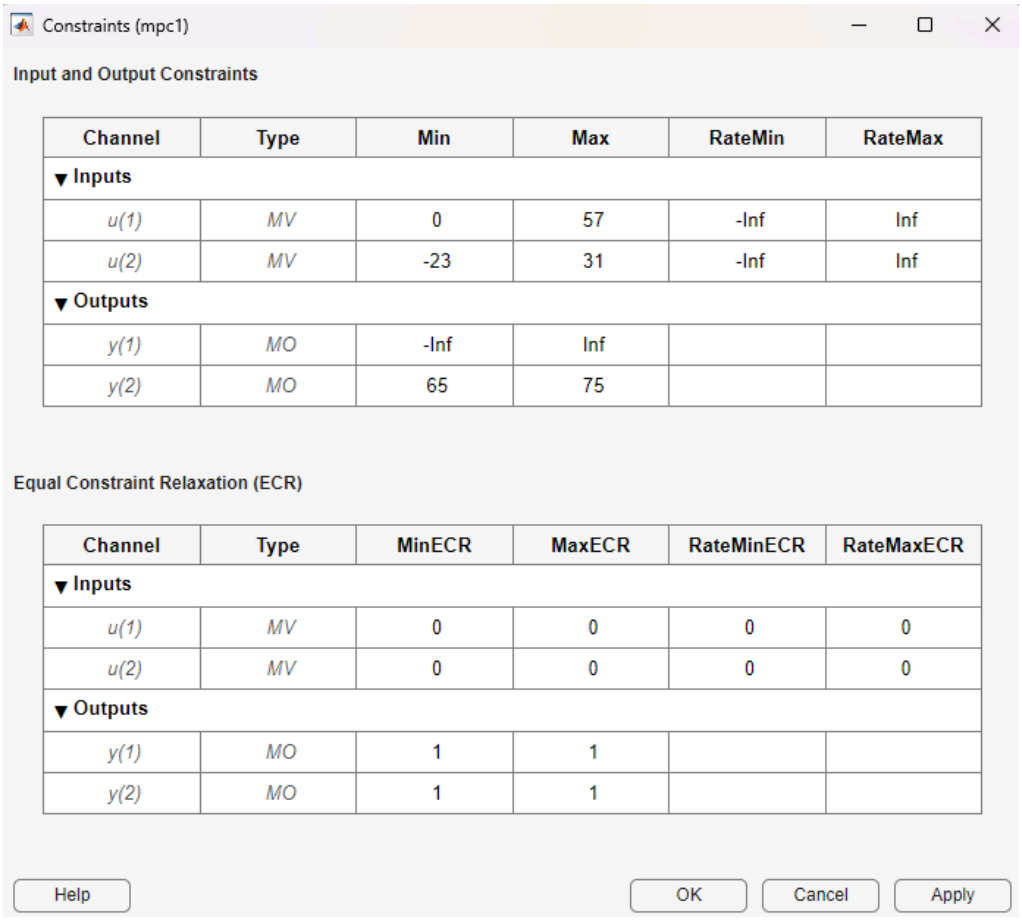
	Channel	Type	Name	Unit	Nominal Value	Scale Factor
1	u(1)	MV	Peng	KW	0	57
2	u(2)	MV	Pbatt	KW	0	54
3	u(3)	MD	Proad	KW	0	1

Plant Outputs

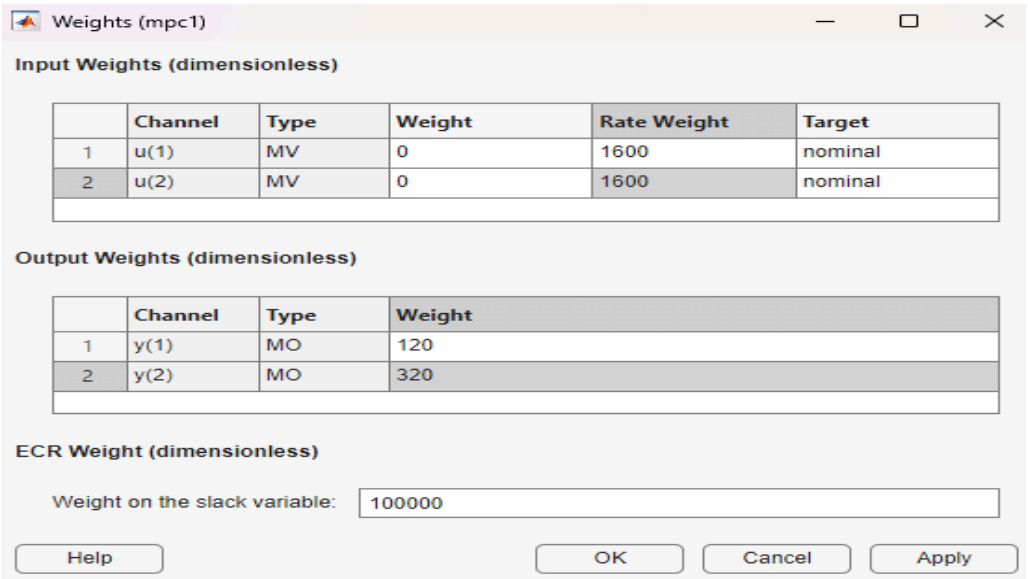
	Channel	Type	Name	Unit	Nominal Value	Scale Factor
1	y(1)	MO	Q	J	0	1
2	y(2)	MO	SOC	%	0	1

Help OK Cancel Apply

These are the input/output channel settings where we specify units as well as scale factors for Peng and Pbatt.



Above is the tab of constraints specifying for the UDDS drivecycle.



In this tab we give weights to individual variables stating 1600 for both control inputs and 120 for Q output whereas 320 for the SOC variable. I got these values by tuning a little to get all required constraints and conditions.

Control signal :

The MPC controller utilizes a MIMO framework i.e multi input and multi output framework and in our project for a power split Hybrid electric vehicle we are using three inputs and 2 outputs.

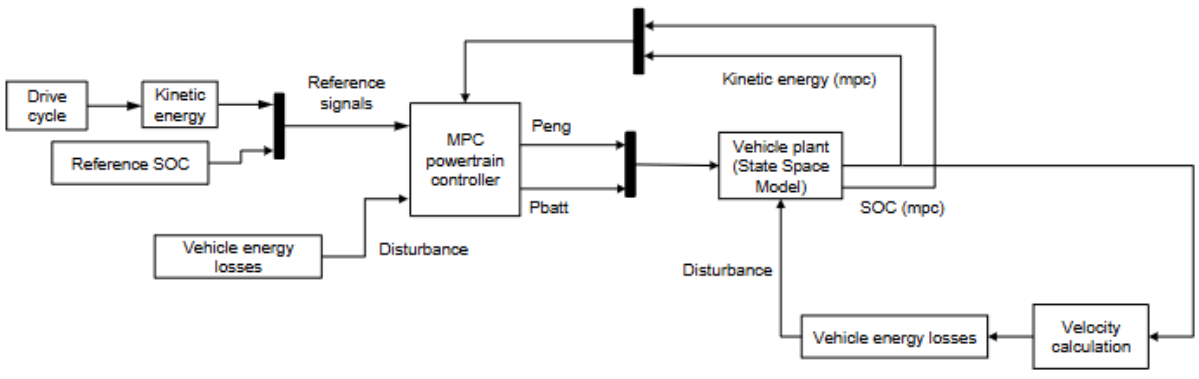
Control Inputs :

1. **Engine power (Peng)** - a manipulated variable which gives us power from Engine.
2. **Battery power (Pbatt)** - a manipulated variable that gives us power from the battery or regeneration.
3. **Road load power (Proad)** - a measured disturbance which is based on vehicle velocity, aerodynamic drag, rolling resistance and road grade.

System Outputs :

1. **Kinetic energy (Q)** - We compute this with vehicle mass and velocity and it represents the power demand.
2. **State of charge (SOC) -** Indicates the current battery charge level.

Working principle:
The controller is provided with reference velocity and SOC values based on the drive cycle.
It also takes into account true losses like road grade and drag.
With this, the MPC calculates the amount of power to pull from the engine and battery.
The vehicle model changes its state and returns Q and SOC for the next control step.



Standard Cost Function :-

$$J = \sum_{j=1}^{n_y} \sum_{i=1}^{N_p} \left\{ w_j^y \left[r_j(k+i/k) - y_j(k+i/k) \right]^2 \right\} + \sum_{j=1}^{n_u} \sum_{i=0}^{N_c-1} \left\{ w_j^{\Delta u} \left[u_f(k+i/k) - u_f(k+i-1/k) \right]^2 \right\} + \sum_{j=1}^{n_u} \sum_{i=0}^{N_c-1} \left\{ w_j^u \left[u_f(k+i/k) - u_{j,target}(k+i/k) \right]^2 \right\} + \int_{\varepsilon} \varepsilon_k^2$$

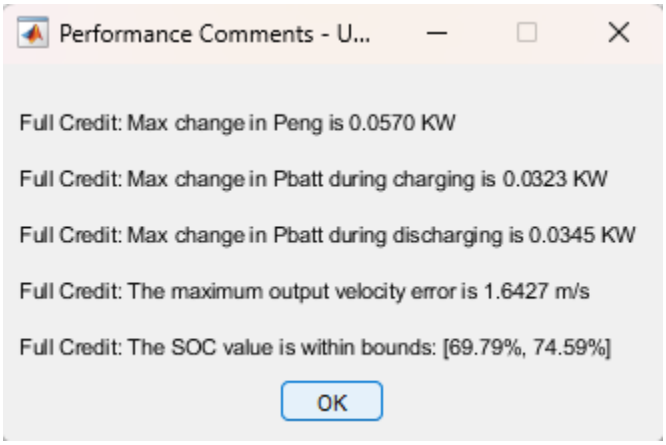
Term 1 measures output reference tracking.
Term 2 adjustments of control input values.
Term 3 measures if control inputs are at/near specified target values
Term 4 is constraint violation for soft constraints.

Our Cost Function is :-

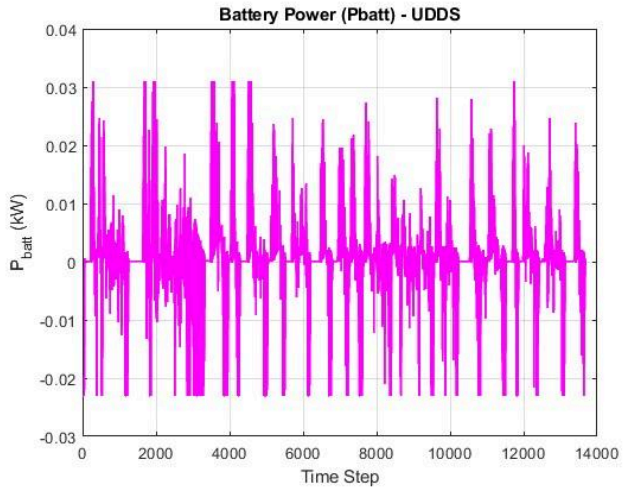
$$J = \sum_{i=1}^{N_p} \left\{ w_Q (Q_{ref} - Q_{out})^2 \right\} + \sum_{i=1}^{N_p} \left\{ w_{soc} (soc_{ref} - soc_{out})^2 \right\} + \sum_{i=0}^{N_i-1} \left\{ w_{batt} (\Delta P_{batt})^2 \right\} + \sum_{i=0}^{N_i-1} \left\{ w_{eng} (\Delta P_{eng})^2 \right\} + \sum_{i=1}^{N_p} \left\{ w_v (V_{ref} - V_{out})^2 \right\}.$$

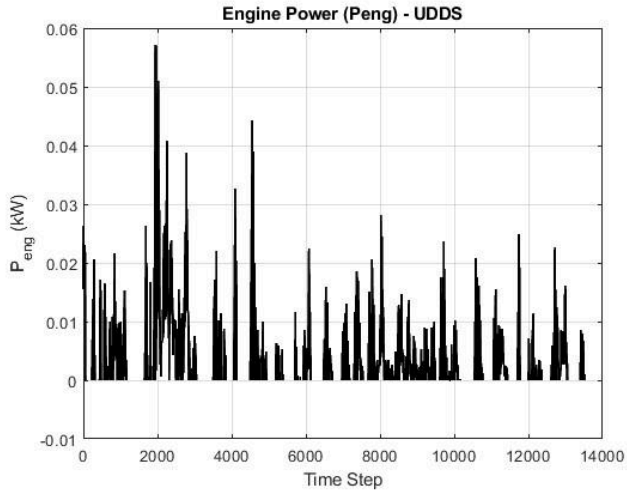
UDDS DRIVE CYCLE :

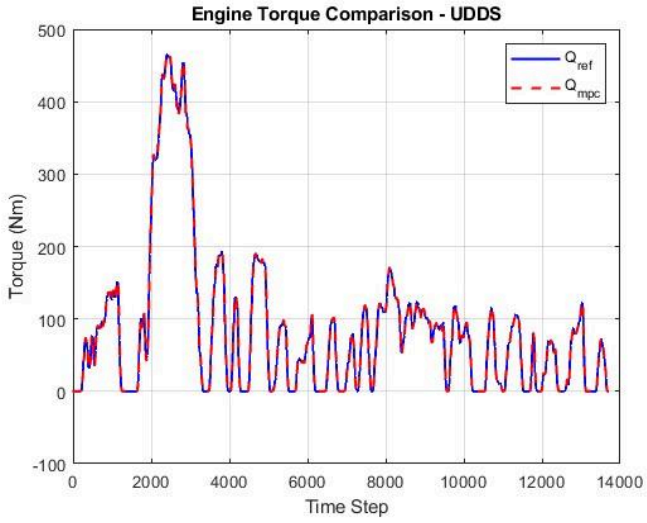
SIMULATION RESULTS :

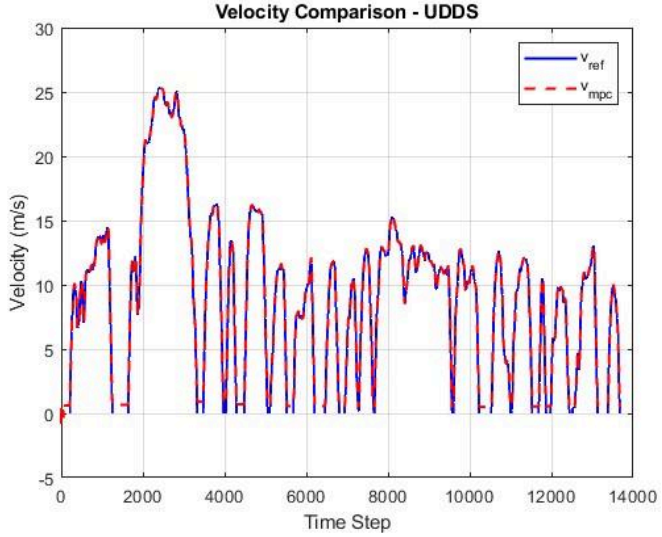


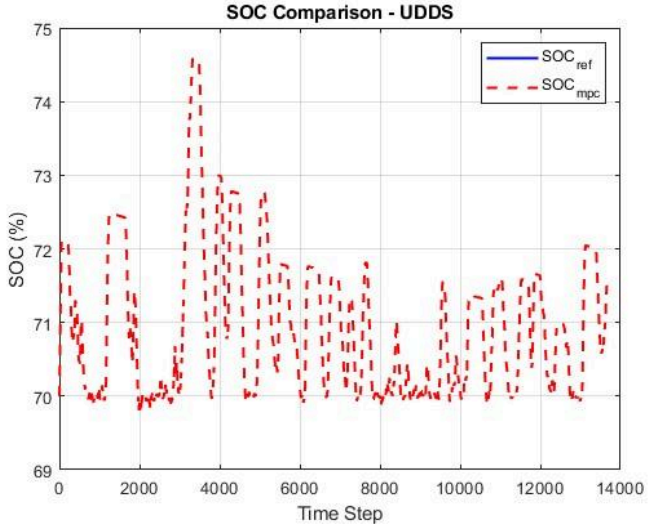
Successfully got through all conditions by running the mpc controller designed under UDDS drive cycle.






Engine Torque Comparison - UDDS
Torque (Nm)
Time Step
Q_ref
Q_mpc


Velocity Comparison - UDDS
Velocity (m/s)
Time Step
V_ref
V_mpc


SOC Comparison - UDDS
SOC (%)
Time Step
SOC_ref
SOC_mpc

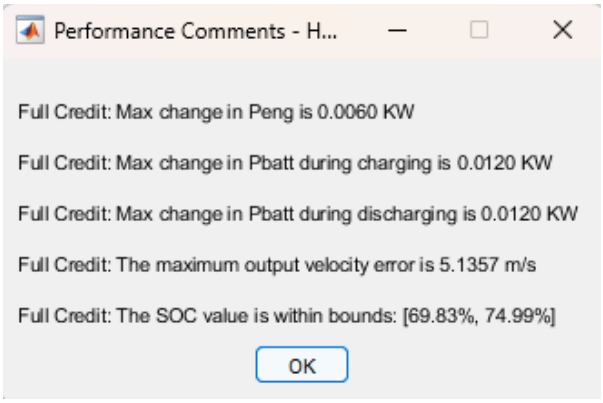
Above are the graphs of battery power, engine power, engine torque, Velocity comparison and the SOC comparison between reference and simulated.

Influence of MPC Design on Performance:

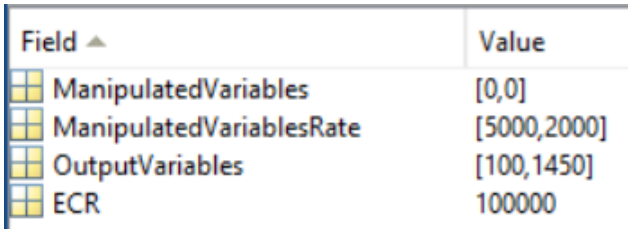
MPC design adjustment modifies the controller's performance-smoothness trade-off. Adding weight to engine torque, made it track more accurately but decreased power changes to less smooth levels. Increasing SOC control emphasis kept the battery within limits but decreased speed tracking slightly. Changing weights on power changes made the response smoother but slower. Generally, each design change modifies the way the car responds.

HWFET DRIVE CYCLE :

For the HWFET Drive cycle we use the same MPC structure and the settings but changed the constraints as per required parameter values and conditions.



We obtained full credits for all the required conditions with our controller for the HWFET Drive cycle. And below are the plots for battery power, engine power, engine torque, velocity comparison and SOC comparison between reference and simulated.



Weights (mpcHEV_HWFET)

Input Weights (dimensionless)

	Channel	Type	Weight	Rate Weight	Target
1	u(1)	MV	0	5000	nominal
2	u(2)	MV	0	2000	nominal

Output Weights (dimensionless)

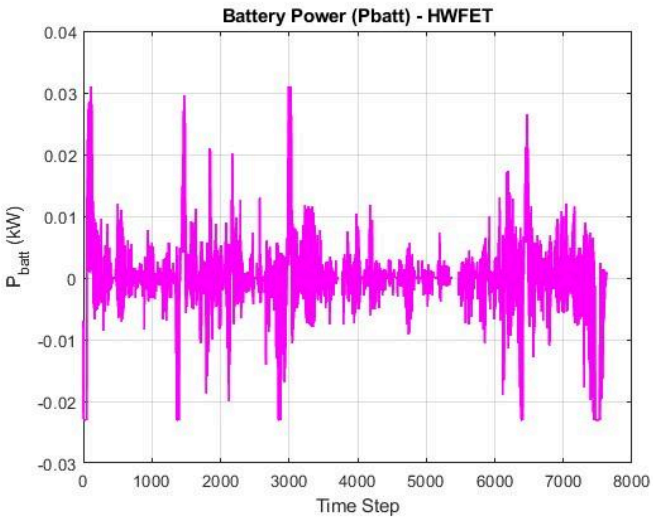
	Channel	Type	Weight
1	y(1)	MO	100
2	y(2)	MO	1450

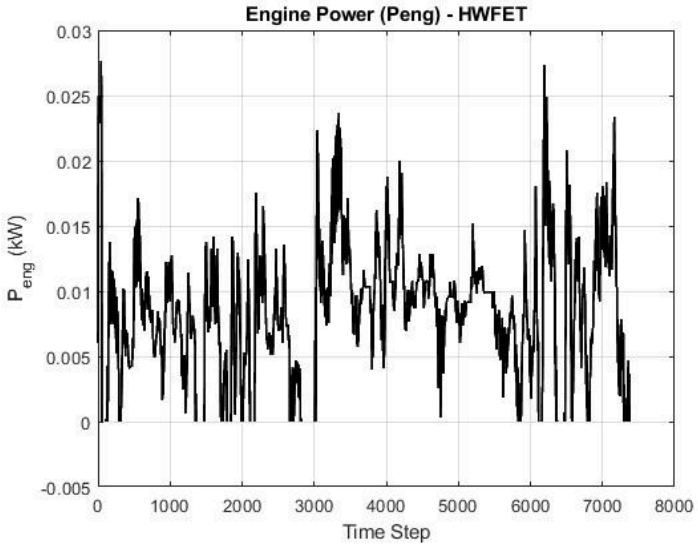
ECR Weight (dimensionless)

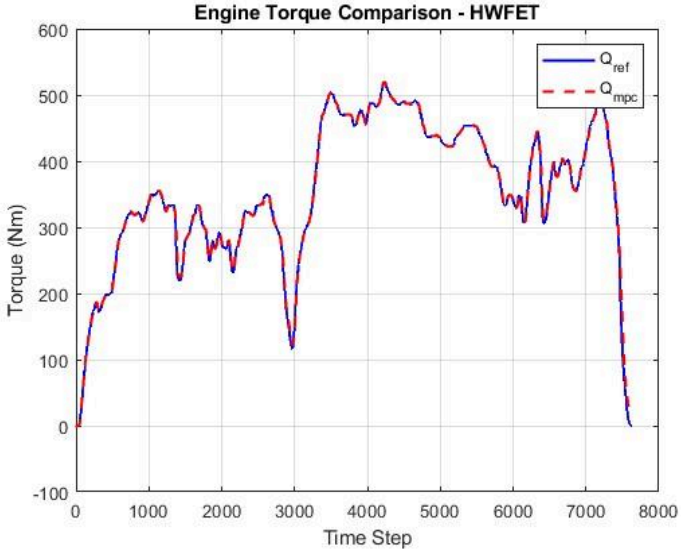
Weight on the slack variable: 100000

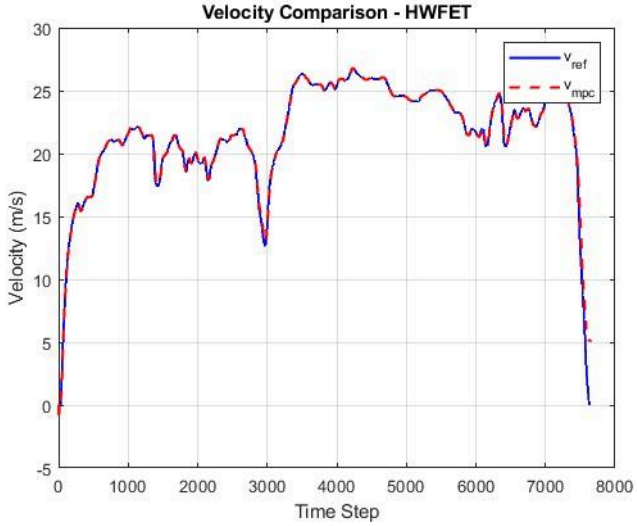
Help OK Cancel Apply

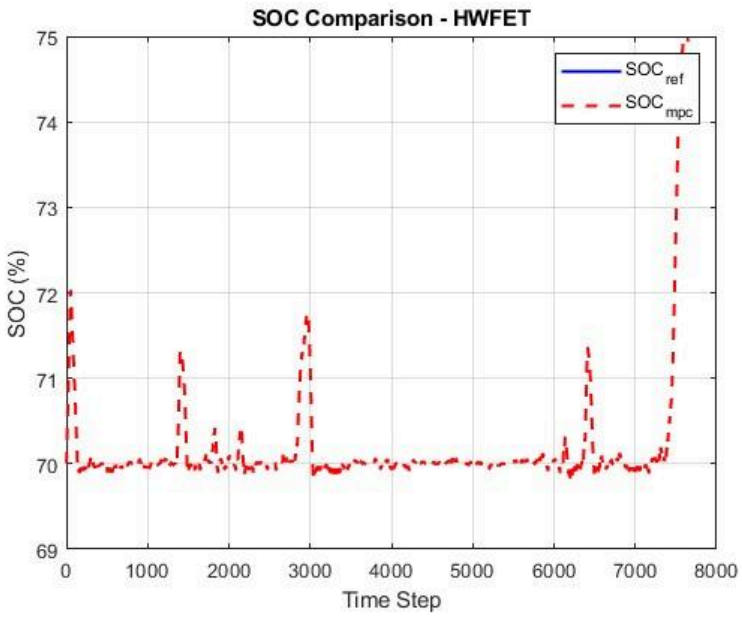
Above are the weights used for the HWFET drive cycle MPC controller. And we obtained plots of battery power, engine power, engine torque, Velocity comparison and the SOC comparison between reference and simulated.




Engine Power (Peng) - HWFET
P_eng (kW)
Time Step
0.03
0.025
0.02
0.015
0.01
0.005
0
-0.005
0
1000
2000
3000
4000
5000
6000
7000
8000


Engine Torque Comparison - HWFET
Torque (Nm)
Time Step
Q_ref
Q_mpc
600
500
400
300
200
100
0
-100
0
1000
2000
3000
4000
5000
6000
7000
8000


Velocity Comparison - HWFET
Velocity (m/s)
Time Step
v_ref
v_mpc
30
25
20
15
10
5
0
-5
0
1000
2000
3000
4000
5000
6000
7000
8000



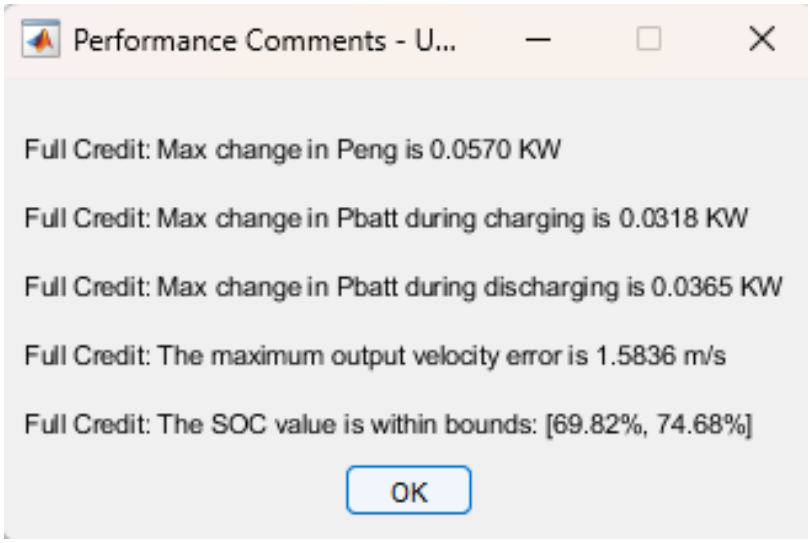
Influence of MPC Design on Performance – HWFET Cycle

For the HWFET cycle, changing the MPC weights noticeably affected how the vehicle behaved. Giving more importance to engine torque made it follow the target better, but caused bigger shifts in battery power. Focusing more on SOC control kept the battery in a safer range, but slightly affected how well the car kept up with the speed profile. Increasing the weight on power smoothness made the ride feel more stable, but the response became a bit slower. Overall, it was all about finding the right balance between efficiency, comfort, and control.

Simulations for four cases with UDDS drive cycle

CASE 1:

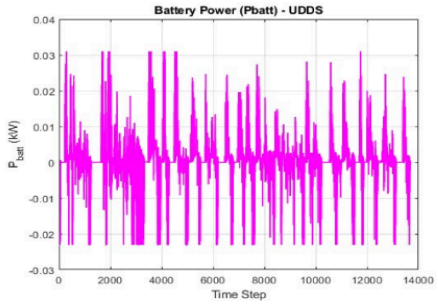
Increasing weight of Q by a factor of 2.



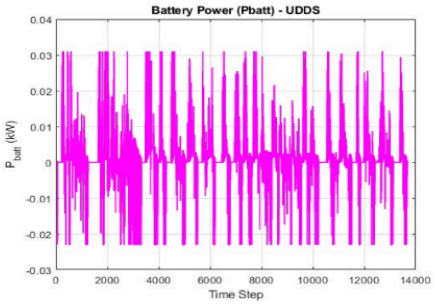
In Case 1, increasing the weight on engine torque helped the controller track the torque reference more precisely. As a result, the battery showed slightly sharper power variations to support this. The SOC still stayed within limits but showed a bit more fluctuation. Velocity tracking remained smooth and accurate. Overall, the system became more responsive in torque control without compromising drive performance.

Comparison of Original UDDS vs Case 1 (Double Q Weight)

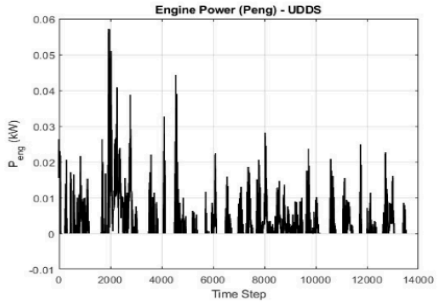
Original UDDS - Battery Power



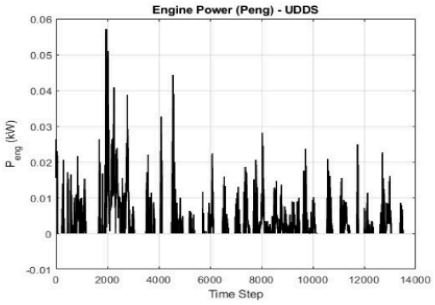
Case 1 - Battery Power



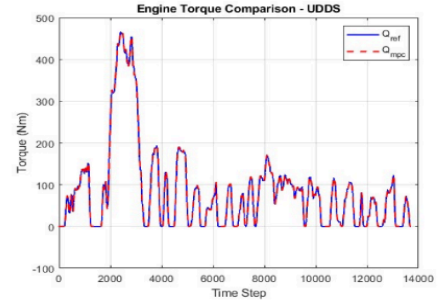
Original UDDS - Engine Power



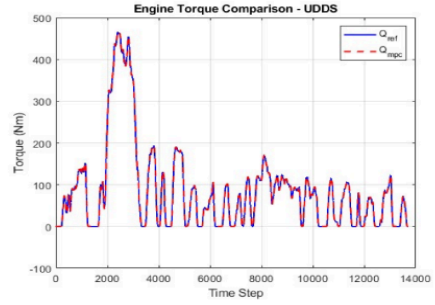
Case 1 - Engine Power



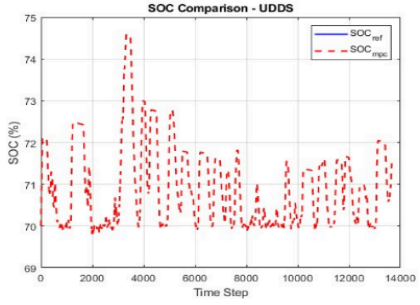
Original UDDS - Engine Torque



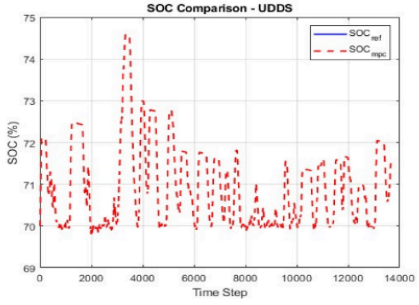
Case 1 - Engine Torque



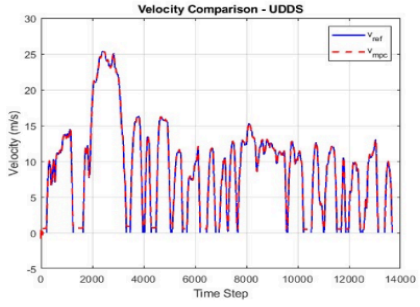
Original UDDS - SOC



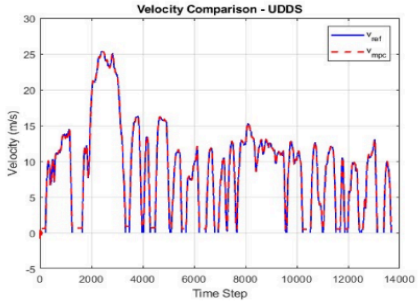
Case 1 - SOC



Original UDDS - Velocity

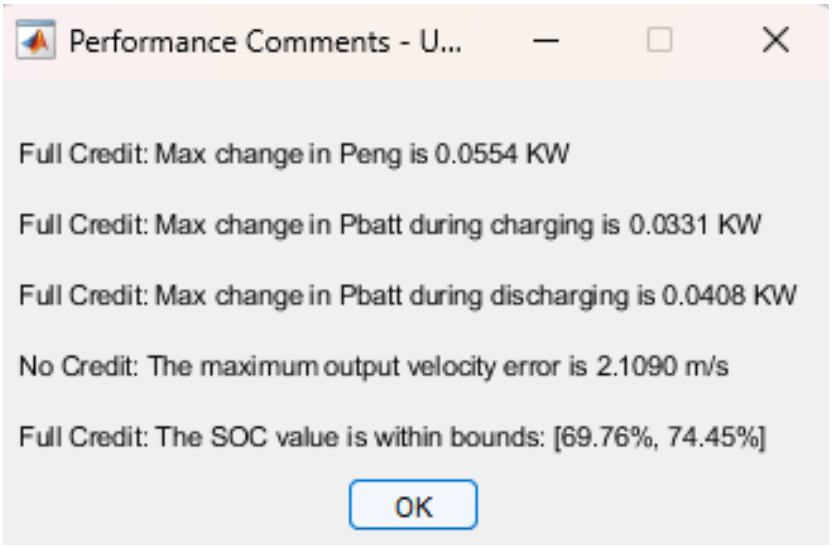


Case 1 - Velocity



CASE 2:

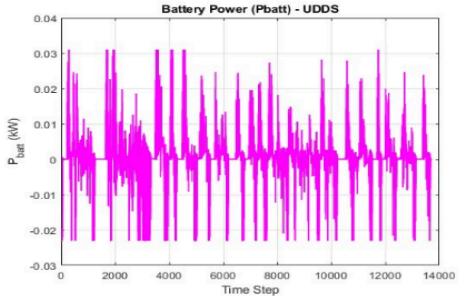
Increasing weight of SOC by a factor of 2.



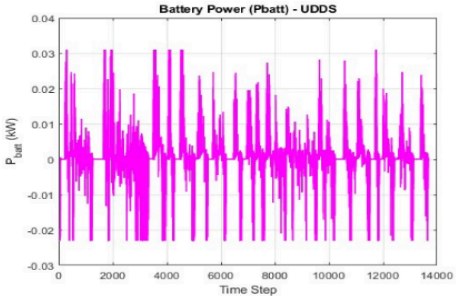
In Case 2, where we doubled the SOC weight, the controller successfully kept the battery SOC well within the target range. All power transitions stayed smooth, and torque was well-managed. The only trade-off was a slight increase in velocity tracking error, only 1.090 m/s more than required constraint.

UDDS vs Case 2 Comparison

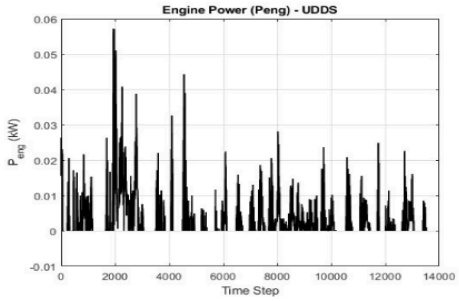
Battery Power - Original



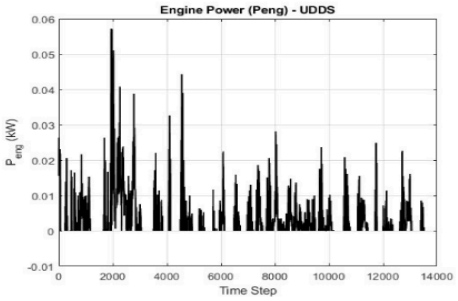
Battery Power - Case 2



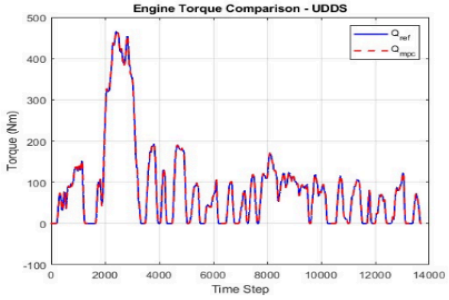
Engine Power - Original



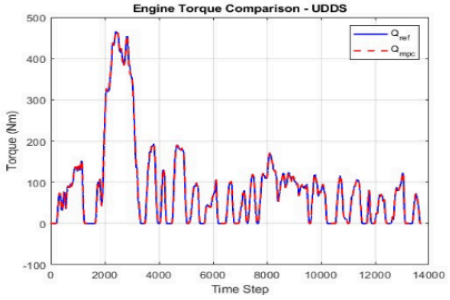
Engine Power - Case 2



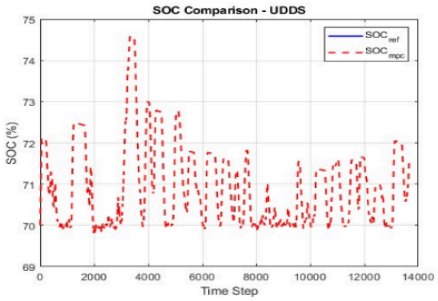
Engine Torque - Original



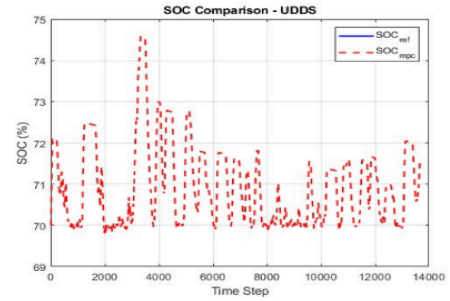
Engine Torque - Case 2



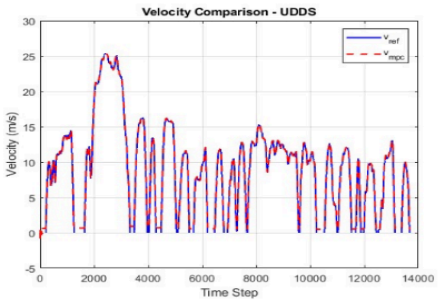
SOC - Original



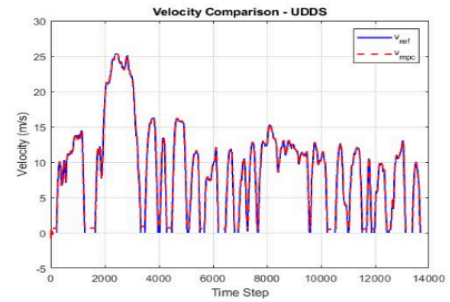
SOC - Case 2



Velocity - Original

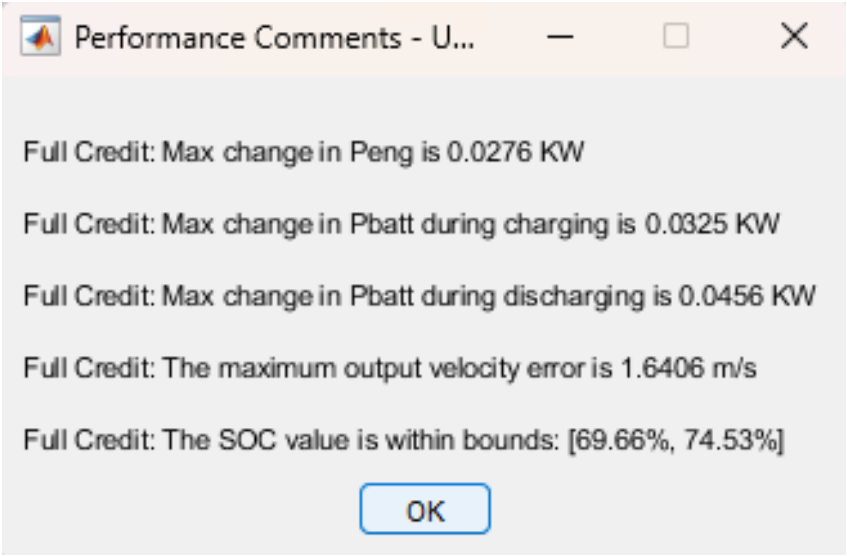


Velocity - Case 2



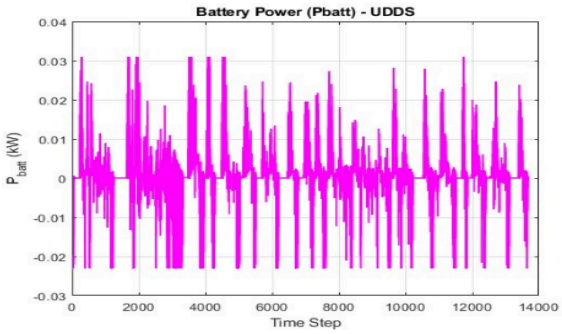
CASE 3:

Increasing weight of Peng by a factor of 2.

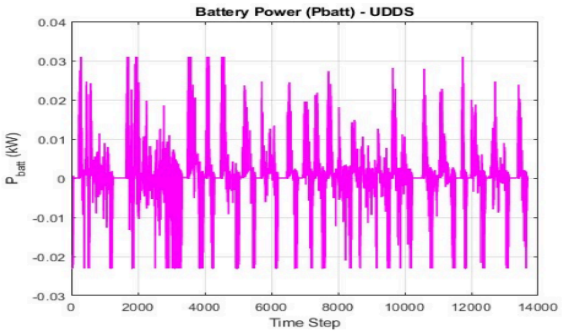


In Case 3, the battery power varies a bit more, likely because the engine was allowed to deliver more power. Compared to the original UDDS, the engine now works more steadily without sharp peaks, showing it's handling the load more smoothly. Even with these changes, the torque and speed follow the targets just as well, which means the control system is still doing a solid job. Interestingly, the SOC stays a little higher in Case 3, hinting that the system is managing energy a bit more efficiently.

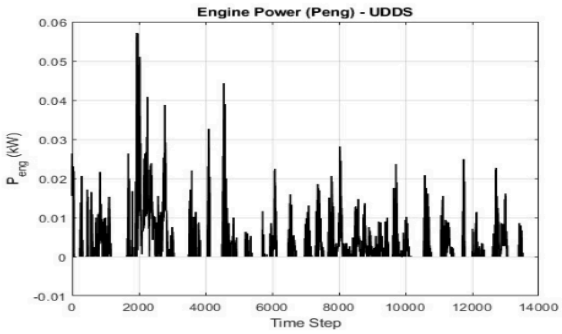
UDDS - Battery Power (Pbatt)



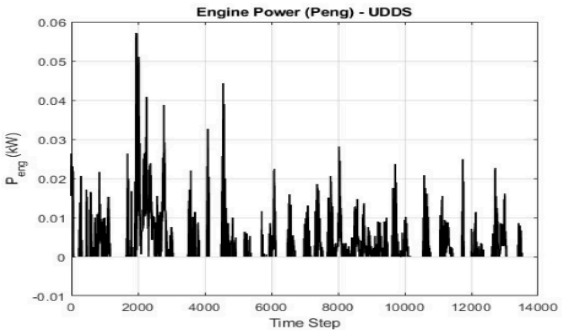
Case 3 - Battery Power (Pbatt)



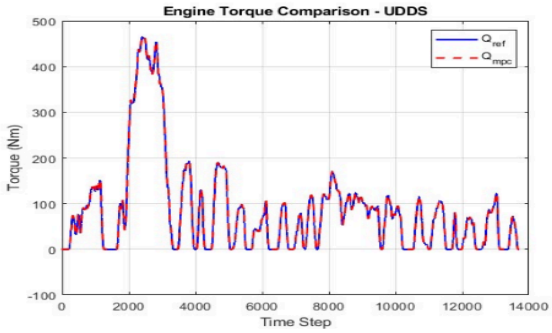
UDDS - Engine Power (Peng)



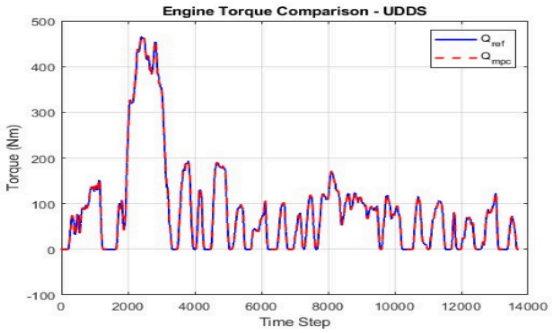
Case 3 - Engine Power (Peng)



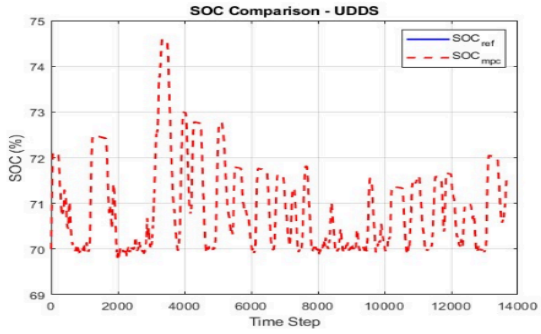
UDDS - Engine Torque



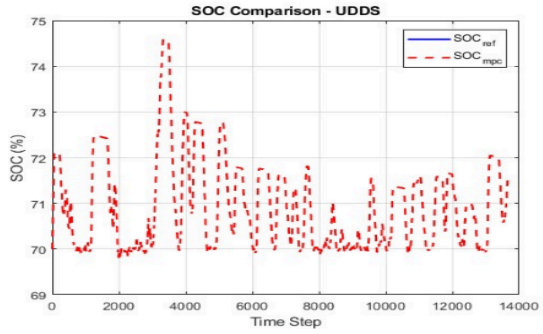
Case 3 - Engine Torque



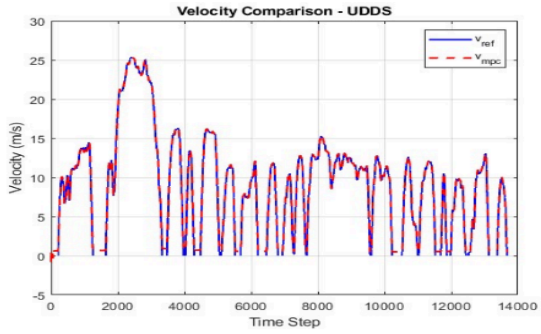
UDDS - SOC Comparison



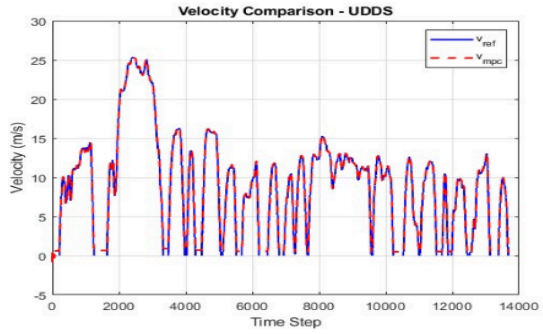
Case 3 - SOC Comparison



UDDS - Velocity Comparison

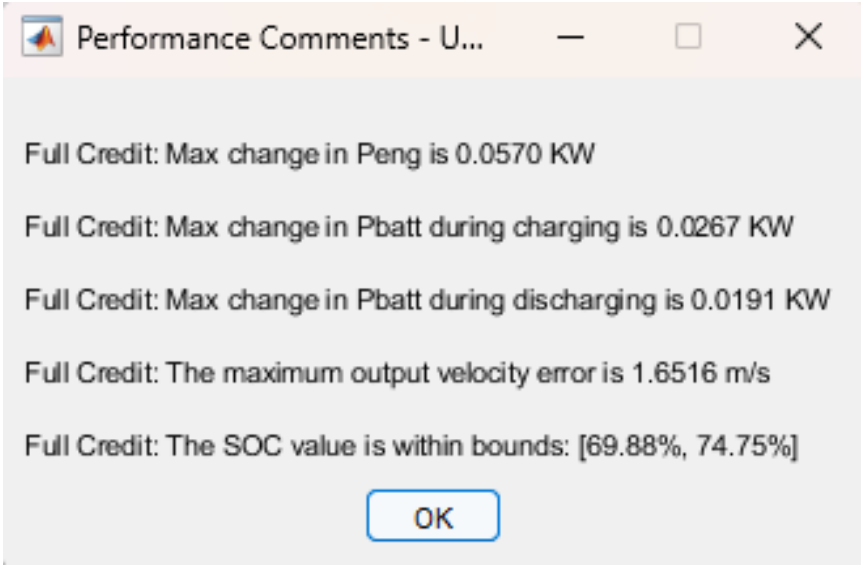


Case 3 - Velocity Comparison



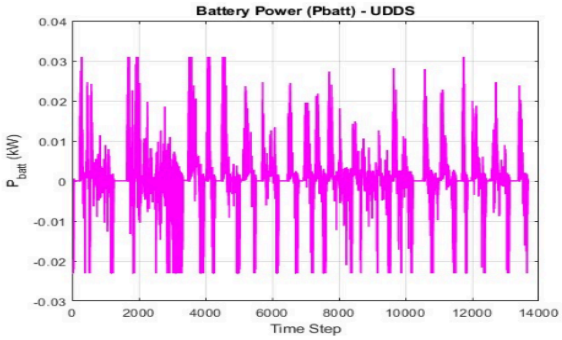
CASE 4:

Increasing weight of Pbatt by a factor of 2.

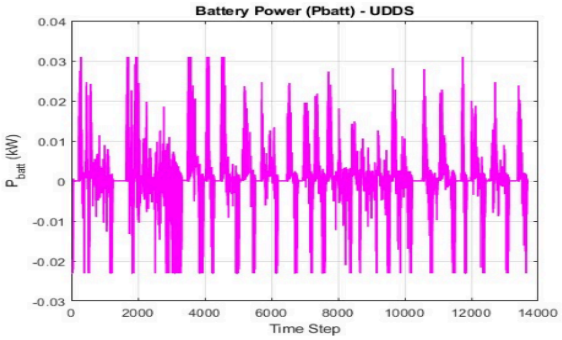


In Case 4, the battery power usage appears slightly smoother but still maintains a similar trend to the original UDDS, suggesting stable support from the battery. The engine power plot also shows a consistent pattern, although the peaks seem a bit more refined compared to the original. Both engine torque and vehicle velocity follow their reference curves very closely, confirming reliable tracking performance. The SOC trajectory in Case 4 remains well within safe limits and mirrors the original behavior, indicating good energy balance without aggressive depletion or overcharging.

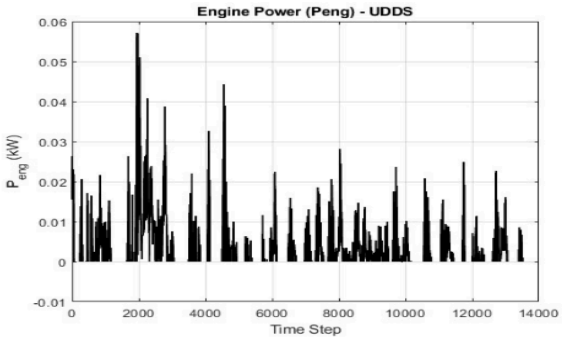
UDDS - Battery Power (Pbatt)



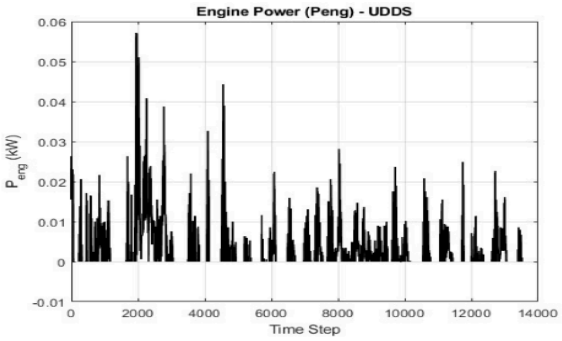
Case 4 - Battery Power (Pbatt)



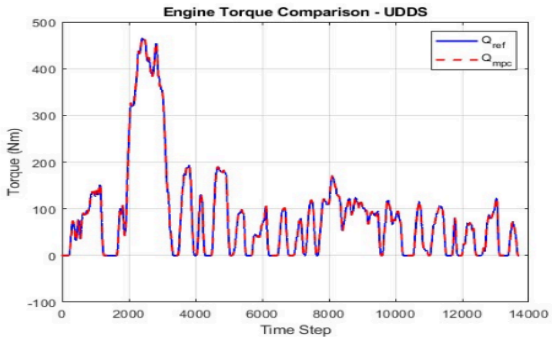
UDDS - Engine Power (Peng)



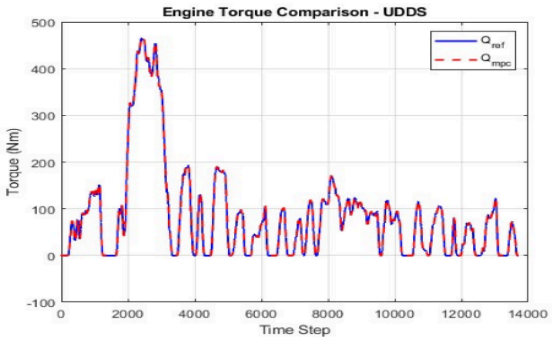
Case 4 - Engine Power (Peng)



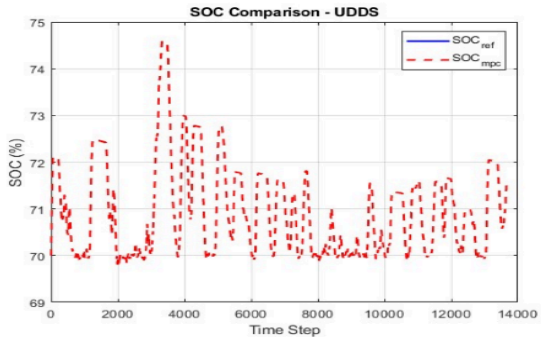
UDDS - Engine Torque



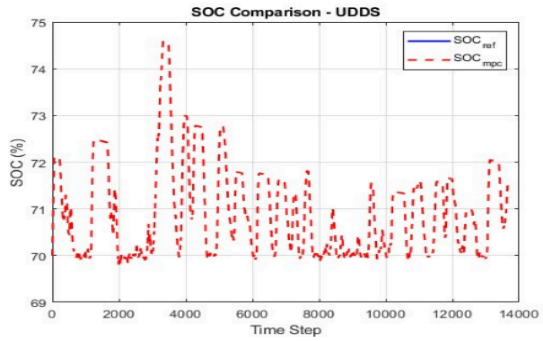
Case 4 - Engine Torque



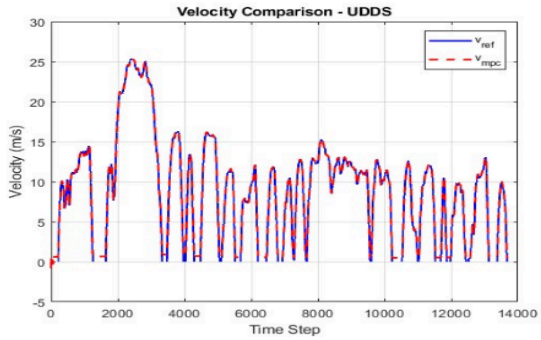
UDDS - SOC Comparison



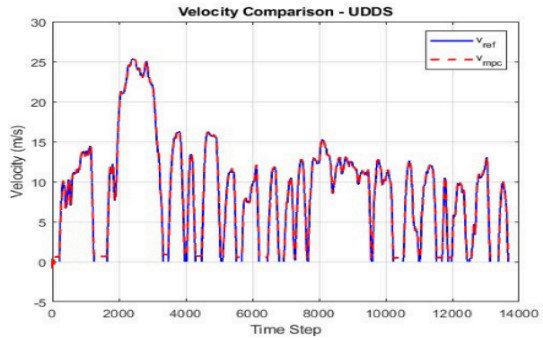
Case 4 - SOC Comparison



UDDS - Velocity Comparison



Case 4 - Velocity Comparison



APPENDIX :

Code for plots and evaluation stopfcn code.

```
out = evalin('base', 'out');
if evalin('base', 'choice_p') == 1
   cycle_name = 'UDDS';
   limit_dPeng = 10000;            % W
   limit_dPbatt_discharge = 5000; % W
   limit_dPbatt_charge = 9500;    % W
   limit_v_error = 2;              % m/s
   soc_range = [65, 75];           % %
else
   cycle_name = 'HWFET';
   limit_dPeng = 6100;
   limit_dPbatt_discharge = 5000;
   limit_dPbatt_charge = 10000;
   limit_v_error = 5.2;
   soc_range = [65, 75];
end
%% PLOT
% 1. SOC_ref vs SOC_mpc
figure;
plot(out.SOC_ref, 'b', 'LineWidth', 1.5); hold on;
plot(out.SOC_mpc, 'r--', 'LineWidth', 1.5);
legend('SOC_{ref}', 'SOC_{mpc}');
xlabel('Time Step'); ylabel('SOC (%)');
title(['SOC Comparison - ' cycle_name]); grid on;
% 2. v_ref vs v_mpc
```

```
figure;
plot(out.V_ref, 'b', 'LineWidth', 1.5); hold on;
plot(out.V_mpc, 'r--', 'LineWidth', 1.5);
legend('v_{ref}', 'v_{mpc}');
xlabel('Time Step'); ylabel('Velocity (m/s)');
title(['Velocity Comparison - ' cycle_name]); grid on;
% 3. Q_ref vs Q_mpc
figure;
plot(out.Q_ref, 'b', 'LineWidth', 1.5); hold on;
plot(out.Q_mpc, 'r--', 'LineWidth', 1.5);
legend('Q_{ref}', 'Q_{mpc}');
xlabel('Time Step'); ylabel('Torque (Nm)');
title(['Engine Torque Comparison - ' cycle_name]); grid on;
% 4. Peng vs Time
figure;
plot(out.Peng / 1000, 'k', 'LineWidth', 1.5);
xlabel('Time Step'); ylabel('P_{eng} (kW)');
title(['Engine Power (Peng) - ' cycle_name]); grid on;
% 5. Pbatt vs Time
figure;
plot(out.Pbatt / 1000, 'm', 'LineWidth', 1.5);
xlabel('Time Step'); ylabel('P_{batt} (kW)');
title(['Battery Power (Pbatt) - ' cycle_name]); grid on;
%% EVALUATION
% 1. Max ΔPeng
dPeng = abs(diff(out.Peng));
Peng_Diff_abs = max(dPeng);
if Peng_Diff_abs <= limit_dPeng
```

```
   Comment_Peng = ['Full Credit: Max change in Peng is ',
num2str(Peng_Diff_abs/1000, '%.4f'), ' KW'];

else

   Comment_Peng = ['No Credit: Max change in Peng is ',
num2str(Peng_Diff_abs/1000, '%.4f'), ' KW'];

end

% 2. Max ΔPbatt - Charging Mode

dPbatt = diff(out.Pbatt);

chg_idx = out.Pbatt(2:end) < 0;

Pbatt_Charge_Max = max(abs(dPbatt(chg_idx)));

if Pbatt_Charge_Max <= limit_dPbatt_charge

   Comment_Pbatt_Charge = ['Full Credit: Max change in Pbatt during
charging is ', num2str(Pbatt_Charge_Max/1000, '%.4f'), ' KW'];

else

   Comment_Pbatt_Charge = ['No Credit: Max change in Pbatt during
charging is ', num2str(Pbatt_Charge_Max/1000, '%.4f'), ' KW'];

end

% 3. Max ΔPbatt - Discharging Mode

dis_idx = out.Pbatt(2:end) > 0;

Pbatt_Discharge_Max = max(abs(dPbatt(dis_idx)));

if Pbatt_Discharge_Max <= limit_dPbatt_discharge

   Comment_Pbatt_Discharge = ['Full Credit: Max change in Pbatt during
discharging is ', num2str(Pbatt_Discharge_Max/1000, '%.4f'), ' KW'];

else

   Comment_Pbatt_Discharge = ['No Credit: Max change in Pbatt during
discharging is ', num2str(Pbatt_Discharge_Max/1000, '%.4f'), ' KW'];

end

% 4. Velocity Error

v_error = abs(out.V_ref - out.V_mpc);

V_error_max = max(v_error);
```

```
if V_error_max <= limit_v_error
   Comment_MaxError = ['Full Credit: The maximum output velocity error
is ', num2str(V_error_max, '%.4f'), ' m/s'];
else
   Comment_MaxError = ['No Credit: The maximum output velocity error is
', num2str(V_error_max, '%.4f'), ' m/s'];
end
% 5. SOC Range Check
SOC_min = min(out.SOC_mpc);
SOC_max = max(out.SOC_mpc);
if SOC_min >= soc_range(1) && SOC_max <= soc_range(2)
   Comment_SOC = ['Full Credit: The SOC value is within bounds: [',
num2str(SOC_min, '%.2f'), '%, ', num2str(SOC_max, '%.2f'), '%]'];
else
   Comment_SOC = ['No Credit: The SOC value is out of bounds: [',
num2str(SOC_min, '%.2f'), '%, ', num2str(SOC_max, '%.2f'), '%]'];
end
%% Performance comments
final_comments = sprintf('%s\n\n%s\n\n%s\n\n%s\n\n%s', ...
   Comment_Peng, ...
   Comment_Pbatt_Charge, ...
   Comment_Pbatt_Discharge, ...
   Comment_MaxError, ...
   Comment_SOC);
msgbox(final_comments, ['Performance Comments - ' cycle_name]);
```