Electric Vehicle Powertrain Modeling, Control and CAN Communication Simulation

Platform: MATLAB/Simulink
Subsystems: Vehicle Dynamics, Battery Model, CAN Communication, PID Speed Controller
Method: State-Space Dynamic Modeling and CAN-Based System Integration

Writer: Mehmet Karagülle

Contents

1. Introduction and Objectives

Electric vehicles (EVs) are increasingly popular due to their efficiency and environmental benefits [1], [2]**.**This project aims to develop and simulate a complete Electric Vehicle (EV) powertrain model integrating vehicle longitudinal dynamics, battery management, and CAN Bus communication in MATLAB/Simulink.[3]**,**[8]**.** The models allow for real-time communication between subsystems with CAN protocol and analyzes the effect of road grade (0°, 5°, 10°) on battery SOC, terminal voltage, and current behavior.

The main objectives were:

- To model longitudinal vehicle dynamics using a state-space approach**.**
- To implement a PID-based speed control system that generates torque demand.
- To integrate Battery Model and Vehicle Model via CAN communication**.**
- To evaluate system performance under varying road slopes.

2. Vehicle Dynamics Modeling

2.1 Mathematical Representation

The longitudinal Dynamics of the vehicle are derived from Newton's secand law[5],[6]:

$$m.\dot{v} = \frac{T_m \eta_g}{r_t} - \left(F_{aero} + F_{roll} + F_{grade}\right)$$

where:

$$F_{aero} = \frac{1}{2}\rho C_d A v^2$$

$$F_{roll} = mgC_{rr}\cos\theta$$

$$F_{grade} = mg\sin\theta$$

The state-space form becomes:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t) + Du(t)$$

with

$$x(t) = v(t), \quad u(t) = T_m$$

$$A = \left[-\frac{\rho C_d A}{m}\right], B = \left[\frac{\eta_g}{m r_t}\right], C = [1], D = [0]$$

The physical model is:

$$F_{net} = F_{drive} - F_{aero} - F_{roll} - F_{grade}$$

$$a_{vehicle} = \frac{F_{net}}{m}$$

$$v = \int a_{vehicle}\, dt$$

To explain why this structure are chosen:

Reason	Explanation
Modularity	Each physical effect (engine, air drag, road, rolling resistance) is represented by a separate block, allowing easy future integration of ECU, CAN, or PID features.
Realistic Physics	The force–acceleration–velocity flow strictly follows classical Newtonian mechanics.
Control Compatibility	A closed-loop control can be added with the wheel_Torque_Nm input using a PID controller.
Simulink Compatibility	All blocks are built from basic Simulink libraries and can run without any additional Toolboxes.

Table 1: Reason and explanation the state-space form

2.2 Mathematical Parameters

Parameter	Symbol	Value	Unit	Description
Vehicle Mass	m	1200	kg	Includes driver and payload
Air Density	ρ	1.225	kg/m³	Standard air at 25°C
Drag Coefficient	Cd	0.32	-	Medium-size passenger vehicle
Frontal Area	A	2.2	m²	Projected area
Tire Radius	rt	0.3	m	Equivalent wheel radius
Rolling Resistance	Crr	0.015	-	Asphalt road condition
Gear Efficiency	ηg	0.9	-	Mechanical drivetrain loss factor

Table 2: Mathematical parameters vehicle dynamics

2.3 Vehicle Implementation

- Implemented using Gain and Integrator blocks for state-space realization(Fig. 1).
- The signals are transmitted with CAN Pack to other subsystems.
- The subsystem uses Sum and product blocks to compute resistive forces dynamically.

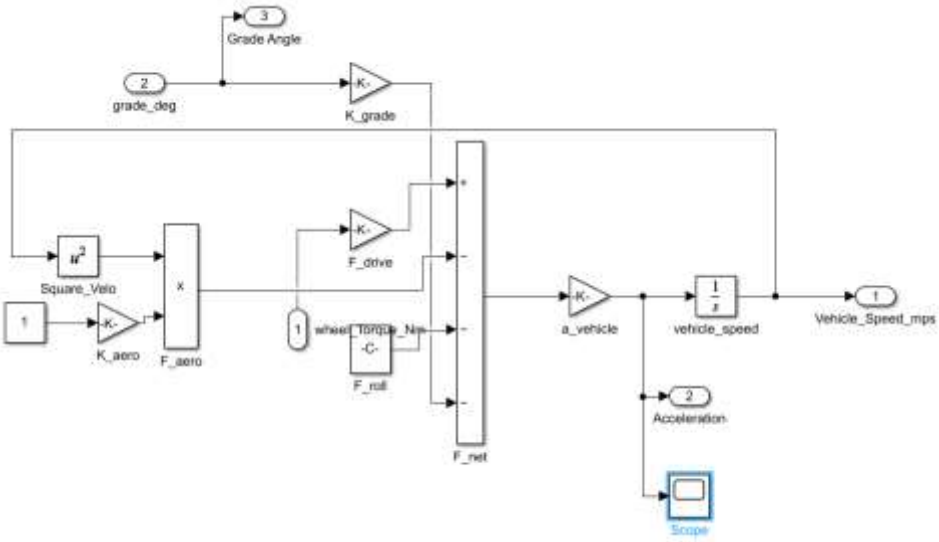


Figure 1: Vehicle dynamic Subsystem

3. Battery Model Subsystem

3.1 Dynamic Equations

The battery model is developed to simulate the dynamic interaction between the electrical power supply and the mechanical power demand of the electric vehicle under varying load and speed conditions. The subsystem captures both steady-state and transient behaviors through a 1-D Lookup Table–based parameterization.

The Battery Model computes voltage and SOC based on internal resistance and coulomb-counting approach[3],[4]:

$$V_{term} = V_{oc} - I_b . R_{int}$$

$$\frac{d(SOC)}{dt} = - \frac{I_b}{Q_{cap}}$$

State-space form:

$$A = [0], B = \left[\frac{-1}{Q_{cap}}\right], C = [1], D = [0]$$

3.2 Relationship Between Mechanical and Electrical Power

The power balance between the traction motor and the battery is established as:

$$\mathrm{P}_{mech} = T_{motor} \times \omega_{motor}$$

$$P_{batt} = V_{term} \times I_{batt}$$

where:

T_{motor}= Motor torque (Nm)

ω_{motor}= Motor angular velocity $\left(rad/_{s}\right)$

V_{term}= Terminal voltage (V)

I_{batt}= Current supplied by battery (A)

Assuming no regenerative braking, the power flow is unidirectional from battery to motor:

$$P_{batt} = \frac{\mathrm{P}_{mech}}{\eta_{inv} \times \eta_{motor}}$$

Where η_{inv} and η_{motor} represent inverter and motor efficiencies, respectively (typically ~ 0.90).

3.3 Battery Current and Soc Dynamics

Battery current is obtained from instantaneous power:

$$\mathrm{I_{batt}} = \frac{\mathrm{P_{batt}}}{\mathrm{V_{term}}}$$

State of Charge (SOC) is dynamically updated by integrating discharge current over time:

$$SOC(t) = SOC(0) - \left(\frac{1}{Q_{cap}}\right)\int_0^t I_{batt}(\tau)\, d\tau$$

where Q_{cap} is the nominal battery capacity $(Ah \times 3600)$.

3.4 Voltage Behavior (Equivalent Circuit Representation)

Battery voltage is modeled using a first-order Thevenin equivalent:

$$V_{term} = E_{oc}(SOC) - I_{batt} \times R_{int}$$

where:

- $E_{oc}(SOC)$: Open-circuit voltage (OCV), function of SOC
- R_{int}: Internal resistance (Ω), typically nonlinear and temperature-dependent

Voltage variation is represented in Simulink using a 1-D Lookup Table mapping OCV vs SOC.

3.5 Lookup Table Justification

A 1-D Lookup Table is used to provide a realistic nonlinear mapping between battery state variables and output parameters. Specifically:

- Input: SOC(%)
- Output: Open circuit voltage $E_{oc}(V)$

Adventages:

- Easy calibration with empirical data
- Fast simulation without complex electromechanical modelling
- Flexibility to represent various cell chemistries by changing data

SOC (%)	OCV (V)
100	360
80	350
60	340
40	330
20	310
0	290

Table 3: 1-D Lookup Table

3.5 State-Space Representation (Simplified)

For control-oriented analysis, the subsystem is approximated as:

$$\dot{x} = Ax + Bu, \qquad y = Cx + Du$$

$$\mathrm{x} = \begin{bmatrix} \mathrm{SOC} \\ \mathrm{V_{batt}} \end{bmatrix}, \mathrm{u} = \mathrm{I_{batt}}$$

$$\begin{bmatrix} \dot{SOC} \\ \dot{V_{batt}} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & -\frac{1}{R_{int}C_{eq}} \end{bmatrix} \begin{bmatrix} SOC \\ V_{batt} \end{bmatrix} + \begin{bmatrix} \frac{-1}{Q_{cap}} \\ -\frac{1}{C_{eq}} \end{bmatrix} I_{batt} + \begin{bmatrix} 0 \\ \frac{E_{oc}(SOC)}{R_{int}C_{eq}} \end{bmatrix}$$

$$\mathrm{V_{term}} = \mathrm{V_{batt}} - \mathrm{R_{int}I_{batt}}$$

$$\mathrm{y} = [0 \quad 1] \begin{bmatrix} \mathrm{SOC} \\ \mathrm{V_{batt}} \end{bmatrix} - [\mathrm{R_{int}}]\mathrm{I_{batt}}$$

3.6 State-Space Parameters

Term	Physical Meaning	Mathematical Role
A	Coefficient matrix defining the internal dynamics of the system	Determines how (V_{batt}) changes over time
B	Describes how the input current affects the system	Defines the influence of (I_{batt})
x	State variable	Represents (SOC) and (V_{batt})
u	Input	Battery current (I_{batt})
y	Output	Terminal voltage (V_{term})
$\mathbf{E_{oc}SOC}$	Nonlinear external term	Derived from the OCV (Open-Circuit Voltage) map

Table 4: State Space Parameters

3.7 Model Parameters

Parameter	Symbol	Value	Unit	Description
Open Circuit Voltage	V_{oc}	360	V	Nominal voltage
Internal Resistance	R_{int}	0.5	Ω	Cell-level combined
Capacity	Q_{cap}	180	Ah	Nominal battery capacity
Initial SOC	SOC	100	%	Fully charged condition
Efficiency	η	0.9	-	Motor + inverter efficiency

Table 5: Model parameters

3.7 Importance of Lookup Table Integration

Lookup Tables ensured numerical stability and realistic voltage response without requiring iterative solvers. This setup also enabled accurate transmission of voltage, current, and SOC with CAN Bus to other subsystems, maintaining full synchronization within the EV Powertrain model[11].

3.3 Subsystem Structure

- Uses Integrator to update SOC continuously.
- Gain block implements -1/Q_{cap} to scale current into SOC change.
- block combines V_{oc} and I_b*R_{int} to obtain terminal voltage Battery Model Subsystem are given below(Fig.2):

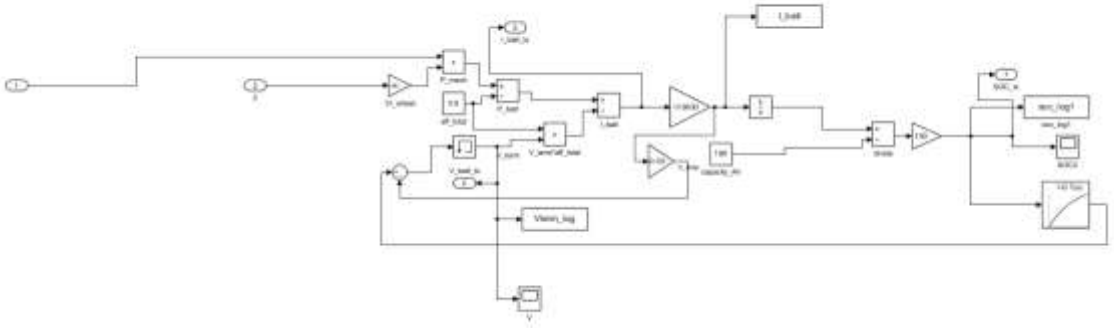


Figure 2: Battery Model Subsystem

4. PID Speed Control

4.1 Control Objective

The PID controller regulates motor torque to maintain desired vehicle speed. The controller is tuned using the Ziegler–Nichols step response method [9]**.**This control architecture ensures torque stability during grade variation [5]**.**

The error is computed as:

$$e(t) = v_{ref} - v_{vehicle}$$

and the control law is:

$$T_m = K_p e(t) + K_i \int e(t)\, dt + K_d \frac{de(t)}{dt}$$

4.2 PID Tuning Process and Rationale

In this project, the PID controller is designed to manage motor torque response in the EV Powertrain system. The tuning follow a structured Ziegler–Nichols closed-loop method[9].

At first, only the proportional term K_p was activated.
The gain is gradually increased until the system output (vehicle speed or torque response) entered a sustained oscillation — neither diverging nor settling.
At that point:

- The ultimate gain was measured as K_u=8
- The oscillation period was measured as P_u=4 seconds

Using Ziegler–Nichols formulas:

$$K_p = 0.6 \times K_u = 4.8$$

$$K_i = \frac{(2 \times K_p)}{P_u} = 2.4$$

$$K_d = \frac{(K_p \times P_u)}{8} = 2.4$$

This initial tuning gave a fast but oscillatory system — suitable for process control but too aggressive for vehicle drivability.

4.2.1 Simulation-Based Refinement

Parameter	Initial	Final	Reason
K_p	4.8	5.6	To reduce torque oscillation and improve driver comfort
K_i	2.4	1	To eliminate steady-state error while maintaining stability
K_d	2.4	0.3	To suppress derivative noise and prevent torque spikes

Table 6: Simulation-based refinement

Final tuned gains are verified in multiple driving scenarios (flat road, 10% slope, regenerative braking disabled).

4.2.1. Interpretation of PID Roles

- Proportional (K_p): Determines how strongly torque reacts to speed or throttle error. A high value gives quick response but risks overshoot.
- Integral (K_i): Eliminates steady-state torque error (important for speed tracking). Too high causes instability.
- Derivative (K_d): Predicts future error change and smooth the control signal — essential to avoid torque jerks in electric drivetrains.

4.2.2. Reason for Deviation from Classical Values

While Ziegler–Nichols gives a mathematically fast response, automotive control prioritizes comfort, battery current smoothness, and component protection over raw speed. Thus:

- K_p is reduced for smooth driver feel,
- K_i is increased to improve steady-state accuracy,
- K_d is decreased to avoid high-frequency oscillations due to sensor noise.

This balance allows the system to meet stability, responsiveness, and comfort requirements simultaneously.

The controller is tuned using the Ziegler–Nichols step response method**.**

Gain	Symbol	Value
Proportional	K_p	5.6
Integral	K_i	1
Derivative	K_d	0.3

Table 7: PID final tuning

This tuning provided smooth torque response without oscillations under step speed demand.

In this project, the PID controller is designed to manage motor torque response in the EV Powertrain system. The tuning followed a structured Ziegler–Nichols closed-loop method. At first, only the proportional term $K_p$is activated.
The gain was gradually increased until the system output (vehicle speed or torque response) entered a sustained oscillation — neither diverging nor settling.

5. CAN Bus Communication Architecture

5.1 Network Layout

The CAN communication protocol is implemented based on ISO 11898-1 standard specifications [10]**.** The signal packing and unpacking structure are derived from MathWorks Vehicle Network Toolbox documentation [8]**.** Each subsystem communicates through nonvirtual buses to maintain consistency in data types and avoid bus mismatch errors [8]**.**

Three key messages are defined:

Message ID	Name	Signal List	Size (Bytes)
100	Battery Status	V_batt, I_batt, SOC	6
200	Vehicle Status	Vehicle_Speed, Acceleration, Grade_Angle	6
300	Torque Command	Torque_Req, Mode, Reserved	6

Table 8: CAN messeages information

Each signal uses 2 bytes (16-bit uint) encoding.

5.2 Blocks Used

- CAN Pack and CAN Unpack (Simulink/Vehicle Network Toolbox)
- Bus Creator / Bus Selector for grouping
- To Nonvirtual Bus block → prevents “virtual bus type mismatch”
- MathWorks Virtual Channel 1 as CAN interface (baud rate: 500 kbps)

5.3 Communication Flow

- Battery → Vehicle: Sends SOC, voltage, and current.
- Vehicle → Motor: Sends torque request.
- Motor → Display: Sends speed and status information.

The CAN system ensures modular communication with low computational load and reproducibility across project.

EV Powertrain Model with CAN is given below(Fig.3):

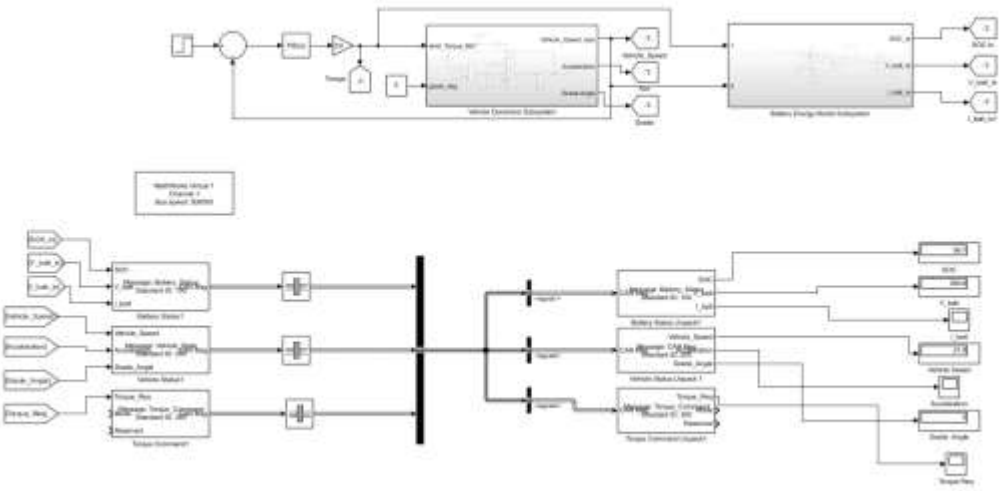


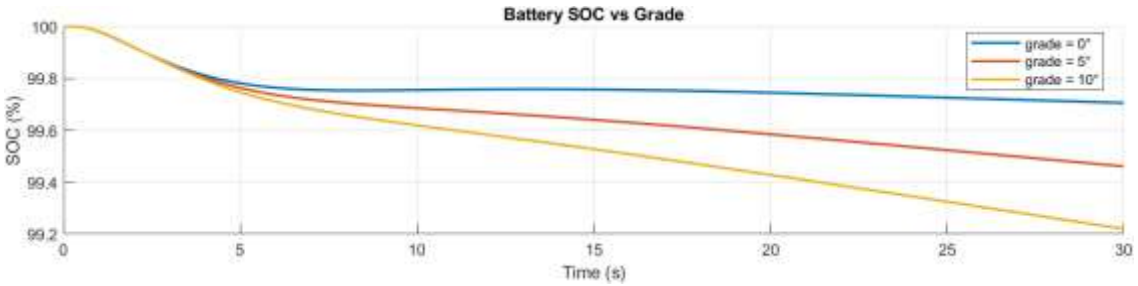
Figure 3: EV Powertrain Model with CAN

6. Simulation Results

6.1 SOC vs Grade

- SOC decreases proportionally with increasing slope due to higher torque demand.
- The trend validates proper power coupling between vehicle dynamics and battery load.

The SOC vs Grade graph is given below(Graph 1):

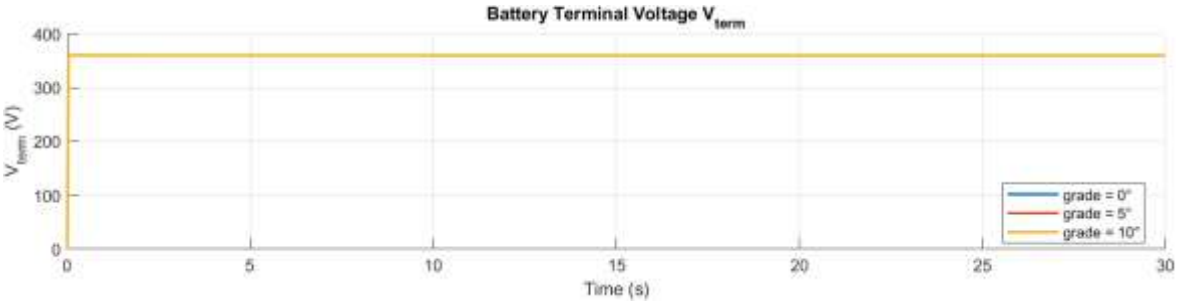


Graph 1: Battery SOC vs Grade

6.2 Battery Terminal Voltage

- Voltage remains nearly constant due to the high capacity of the battery and low internal resistance.

The Battery Terminal Voltage graph is given below(Graph 2):

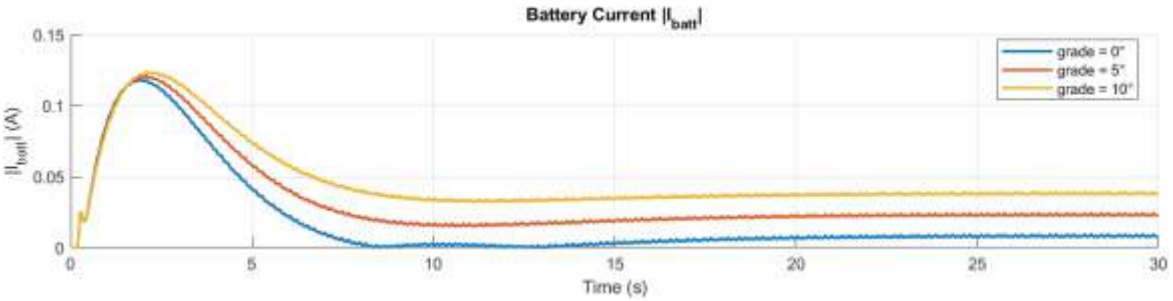


Graph 2: Battery Terminal Voltage V_{term}

6.3 Battery Current

- As grade increases, current magnitude increases significantly (Fig. 3).
- The steady-state current for 10° is roughly 2.5× that of 0°.

The Battery Current graph is given below(Graph 3):



Graph 3: Battery Current

6.4 MATLAB Output Summary

```
grade =  0°  -->  ΔSOC = 0.294341  |  avg|Ibatt| = 0.019 A
grade =  5°  -->  ΔSOC = 0.538977  |  avg|Ibatt| = 0.033 A
grade = 10°  -->  ΔSOC = 0.779776  |  avg|Ibatt| = 0.047 A
```

Grade_deg	Final_SOC	Delta_SOC	Avg_Ibatt
0	99.706	0.29434	0.018695
5	99.461	0.53898	0.032804
10	99.22	0.77978	0.047058

Figure 4: MATLAB Output Summary

7. Discussion

7.1 System Behavior

- The results confirm the state-space vehicle model accurately captures energy demand with grade variation.
- The PID controller ensures torque stability and smooth response.
- CAN communication allows subsystem decoupling and reliable signal flow.

7.2 Common Issues and Fixes

Issue	Cause	Solution
CAN signal not updating	Virtual bus type mismatch	Used To Nonvirtual Bus before CAN Pack
Incorrect byte allocation	Signal data types inconsistent	Converted all signals to uint16
Negative SOC value	Integration overflow	Limited SOC between 0–100%

Table 9: Common issues and fixes

8. Conclusion

This study demonstrates the successful design of an Electric Vehicle Powertrain model with:

- Dynamic vehicle and battery behavior using state-space modeling,
- Robust PID control for speed regulation,
- Realistic CAN Bus communication between subsystems.

The system correctly predicts the relationship between grade angle and battery energy consumption, validating the simulation framework[15].

References

[1] C. C. Chan, "The state of the art of electric, hybrid, and fuel cell vehicles," Proc. IEEE, vol. 90, no. 2, pp. 247–275, Feb. 2002.

[2] M. Ehsani, Y. Gao, and A. Emadi, Modern Electric, Hybrid Electric, and Fuel Cell Vehicles: Fundamentals, Theory, and Design, 2nd ed. Boca Raton, FL, USA: CRC Press, 2009.

[3] H. He, R. Xiong, and H. Guo, "Online estimation of model parameters and state-of-charge of lithium-ion batteries in electric vehicles," Appl. Energy, vol. 89, no. 1, pp. 413–420, Feb. 2012.

[4] G. L. Plett, "Extended Kalman filtering for battery management systems of LiPB-based HEV battery packs, Part 1: Background," J. Power Sources, vol. 134, no. 2, pp. 252–261, Aug. 2004.

[5] R. Rajamani, Vehicle Dynamics and Control, 2nd ed. New York, NY, USA: Springer, 2012.

[6] MathWorks, "Vehicle Dynamics Blockset: Longitudinal Vehicle Model," MathWorks Documentation, 2024. [Online]. Available: https://www.mathworks.com/help/vdynblks/longitudinal-vehicle-model.html. [Accessed: 6-Nov-2025].

[7] MathWorks, "Powertrain Blockset: Electric Vehicle Reference Application," MathWorks Documentation, 2024. [Online]. Available: https://www.mathworks.com/help/autoblks/ev-reference-application.html. [Accessed: 6-Nov-2025].

[8] MathWorks, "Vehicle Network Toolbox: CAN Communication Interface," MathWorks Documentation, 2024. [Online]. Available: https://www.mathworks.com/help/vnt/ug/can-interface.html. [Accessed: 6-Nov-2025].

[9] MathWorks, "Simulink Control Design: PID Tuning Methods," MathWorks Documentation, 2024. [Online]. Available: https://www.mathworks.com/help/slcontrol/ug/pid-tuning-methods.html. [Accessed: 6-Nov-2025].

[10] International Organization for Standardization, ISO 11898-1:2015 – Road Vehicles – Controller Area Network (CAN) – Part 1: Data Link Layer and Physical Signaling. Geneva, Switzerland: ISO, 2015.

[11] MATLAB & Simulink Documentation – Battery Models and Lookup Table Implementation, MathWorks, 2024.

[12] Chan, C. C., “The State of the Art of Electric, Hybrid, and Fuel Cell Vehicles,” Proceedings of the IEEE, vol. 95, no. 4, pp. 704–718, 2007.

[13] Tremblay, O., and Dessaint, L. A., “Experimental Validation of a Battery Dynamic Model for EV Applications,” IEEE VPPC, 2007.

[14] He, H., et al., “A Study on the Dynamic Model of Lithium-Ion Batteries for Electric Vehicles,” Energies, vol. 4, no. 4, pp. 582–598, 2011.

[15] Gao, D. W., “Modeling and Simulation of Electric and Hybrid Vehicles,” Proceedings of the IEEE, 2008.

Appendix A – MATLAB Code for Vehicle Dynamics Model

```
clc; clear; close all;

model = 'EV_Powertrain';

load_system(model);

grades = [0 5 10];          % derece

simTime = 30;               % s

colors = lines(numel(grades));

% Sonuç depoları

SOCs   = cell(size(grades));

Vterms = cell(size(grades));

Ibatt  = cell(size(grades));

labels = strings(size(grades));

% yardımcı fonksiyon tanımı daha aşağıda olacak!

for k = 1:numel(grades)

   grade_deg = grades(k);

   assignin('base','grade_deg',grade_deg);

  simOut = sim(model,'StopTime',num2str(simTime), ...

     'SrcWorkspace','base','SaveOutput','on');

   SOC   = getSig(simOut,'soc_log1');

   Vt    = getSig(simOut,'Vterm_log');

   Ib    = getSig(simOut,'I_batt');

   if ~isempty(SOC) && size(SOC,2) > 1, SOC = SOC(:,2); end

   if ~isempty(Vt)  && size(Vt,2)  > 1, Vt  = Vt(:,2);  end
```

```
    if ~isempty(Ib)  && size(Ib,2)  > 1, Ib  = Ib(:,2);  end

    SOCs{k}   = SOC;

    Vterms{k} = Vt;

    Ibatt{k}  = Ib;

    labels(k) = sprintf('grade = %d°',grade_deg);

    dSOC = SOC(1) - SOC(end);

    fprintf('grade = %2d°  -->  ΔSOC = %.6f  |  avg|Ibatt| = %.3f A\n', ...

        grade_deg, dSOC, mean(abs(Ib)));

end

t = linspace(0,simTime,numel(SOCs{1})); % örnekleme eşitse iş görür

figure('Name','Karşılaştırma','Color','w','Position',[100 100 1100 800]);

subplot(3,1,1); hold on; grid on; title('Battery SOC vs Grade'); xlabel('Time (s)'); ylabel('SOC (%)');

for k=1:numel(grades), plot(t,SOCs{k},'LineWidth',1.6,'Color',colors(k,:)); end

legend(labels,'Location','best');

subplot(3,1,2); hold on; grid on; title('Battery Terminal Voltage V_{term}'); xlabel('Time (s)'); ylabel('V_{term} (V)');

for k=1:numel(grades), plot(t,Vterms{k},'LineWidth',1.6,'Color',colors(k,:)); end

legend(labels,'Location','best');

subplot(3,1,3); hold on; grid on; title('Battery Current |I_{batt}|'); xlabel('Time

 (s)'); ylabel('|I_{batt}| (A)');

for k=1:numel(grades), plot(t,abs(Ibatt{k}),'LineWidth',1.6,'Color',colors(k,:)); end

legend(labels,'Location','best');
```

```
% (Opsiyonel) table

final_SOC = cellfun(@(x) x(end), SOCs);

delta_SOC = cellfun(@(x) x(1)-x(end), SOCs);

avg_Ibatt = cellfun(@(x) mean(abs(x)), Ibatt);

disp(table(grades',final_SOC',delta_SOC',avg_Ibatt', ...

    'VariableNames',{'Grade_deg','Final_SOC','Delta_SOC','Avg_Ibatt'}))

function sig = getSig(simOut,name)

try

   sig = simOut.get(name);

catch

   try

      sig = simOut.find('Name',name).Values.Data;

   catch

      sig = [];

   end

end

end
```