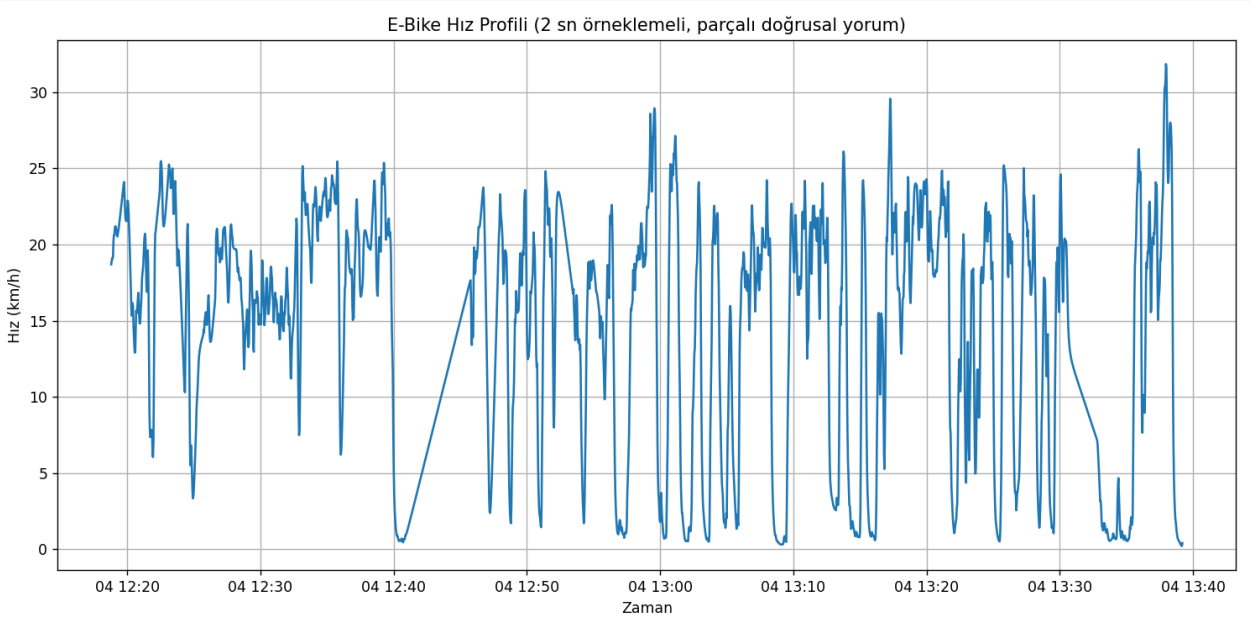
Methods

This study explains the energy consumption, component sizing, and regenerative braking potential of an e-bike under real road conditions. The analysis was carried out using longitudinal dynamic principles and real-world user data. The analysis process was evaluated step by step; instead of calculating all formulas at once, they were first calculated in a straightforward manner and then the system was developed. First, monitoring was performed on the driving cycle and the speed values were examined, then the contribution the user would make to the bicycle was calculated, and afterwards dead zones for regen were defined. After these stages, the actual engineering calculations began and the Kbf constants were identified and how these constants would be applied was defined. Finally, these latest detailed analyses were placed on the efficiency map of the battery and their effect was revealed.

Driving cycle

This is a the raw data used to create driving cycle. To clean and process the chaotic, raw real-world data, we smoothed and downsampled it from 5 Hz to 0.5 Hz(1 data point per 2 second), applied filters, and removed outliers .Our data includes detailed analyses of volunteer drivers recorded in Massachusetts, USA, including their vehicle, age, and average slope. The reason our dataset starts at a fast pace is because the user turns on GPS data while moving. This dataset also includes a wide range of driving styles, with 1.34 hours.



Parameters of Vehicle

In this project, the parameters used are based on the Commuter S250 electric bicycle. We are designing it as a typical "Urban/Trekking E-Bike" that complies with European standards.

Parameter	**Value**	**Rationale**
Total Mass $\boldsymbol{m}$	125kg	Assumption: 100kg rider + 25kg e-bike.
Gravity Acceleration $\boldsymbol{g}$	9.81 m/s²	Standard physical constant.
Rolling Coefficient $\boldsymbol{C_{rr}}$	0.008	Typical tire/road coefficient for asphalt.
Air Density $\boldsymbol{\rho}$	1.225 kg/m³	Standard air density at sea level, 20°C.
Frontal Area $\boldsymbol{A_f}$	0.6 m²	Assumption for Samim Unlusoy. $0.7 \times width(0.5m) \times height(1.71\text{m})$
Drag Coefficient $\boldsymbol{C_D}$	0.75	Aerodynamic shape factor for a semi-recumbent position.
Wheel Radius r_w	0.35 m	Represents a 28-inch (700c) wheel size (≈0.35m radius).
Wind Speed v_{air}	0 m/s	No einf. v_g+v_{air}= v_g
Asistance cons$K_{\text{asistance}}$	0.2	Based on tour mod
Rotary Mass Factorφ	1.1	Typically urban bike
Efficiency Constant n_m	0.88	Direct gear(Samim Unlusoy's note)
Wheelbase	1.15m	Bosch eBike Systems – Engineering Guidelines (2019)
Front–Rear Static Load Distribution	a=0.65 b=0.35	SAE Bicycle Dynamics Paper ve Bosch eBike
CG Height	0.9 m	Bosch eBike Systems, 2019
BMS boundary acceptance	5/30 kph/W 200W	$\frac{Min}{Max}$ regen acceptance

Table 1 - Parameters

Architecture and Controller Design (Hybrid Brake/$K_{asistance}$)

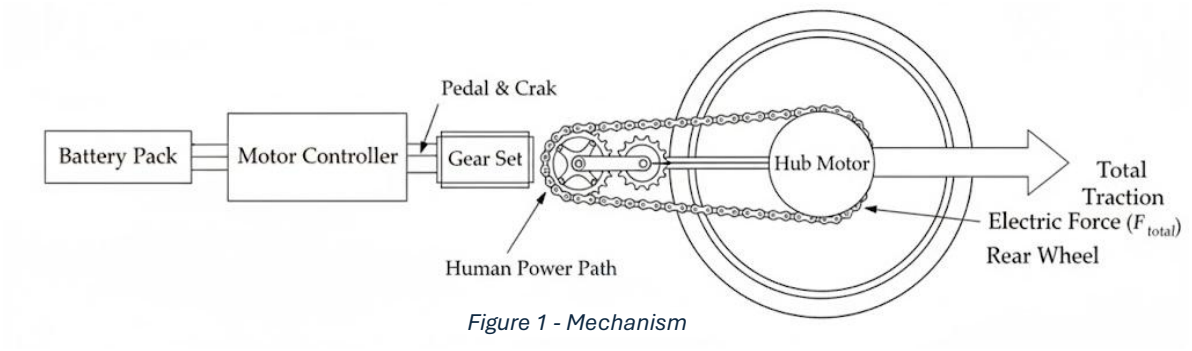


Figure 1 - Mechanism

$$T_h w_h + T_m w_m = T_w w_w$$

$$[w_m = w_w]\ [w_h = i_t w_w]$$

$$T_h i_t + T_m = T_w$$

$$T_{motor} = T_{human} K_{asistance}$$

According to regulations the assistance constant determines how much power our engine should produce. While $P_{motor}(1 + K_{asistance}) = P_{b-out}$ represents the power delivered to the wheel, the expression $\frac{T_m i_t}{K_{asistance}} + T_m = T_w$ represents the torque going to the wheel. Similarly, the power coming from regeneration can be calculated. Below are the forces that prevent us.

Additionally, instead of a conventional mechanical braking mechanism, the braking system is built entirely on regenerative braking on the rear wheel. The main reason for this is that the motors used for e-bikes can generate sufficient counter force for all situations, including emergency breaking. The calculations presented in the following pages will show that even under the most optimal conditions (dry road...) what determines the braking distance is not the braking force itself but wheel slip, in other words the brake force distribution between the wheels. However, the calculations we will perform later will show that most of the energy that could be recovered from breaking is dissipated as heat through the mechanical brakes on the front wheel. Since creating a front-wheel-drive bicycle introduces disadvantages, the braking algorithm will follow Ehsani's rule in which braking demands up to 0.2 g are handled by regenerative braking on the rear wheel; in any other situation, we will distribute braking between the mechanical front brake and the regenerative rear brake using the defined k_{bf} constant.A detailed explanation is provided in the k_{bf} section.

The formulas shown below represent resistive forces for us, meaning that even if the user does not press the brake, the bicycle slows down due to these external effects. These effects have been taken into account in our calculations.

$$F_d = \frac{1}{2} C_d \rho_{air} A \left(v_g + v_{air}\right)^2 \quad \& \quad F_{hill} = Mgsin(\theta) \quad \& \quad F_{rolling} = Mgf_r$$

$$F \times v = P \quad \& \quad F \times r_w = \mathrm{T}$$

Calculation

Analysis of Motor Power: Losses (Traction) & Gains (Regeneration)

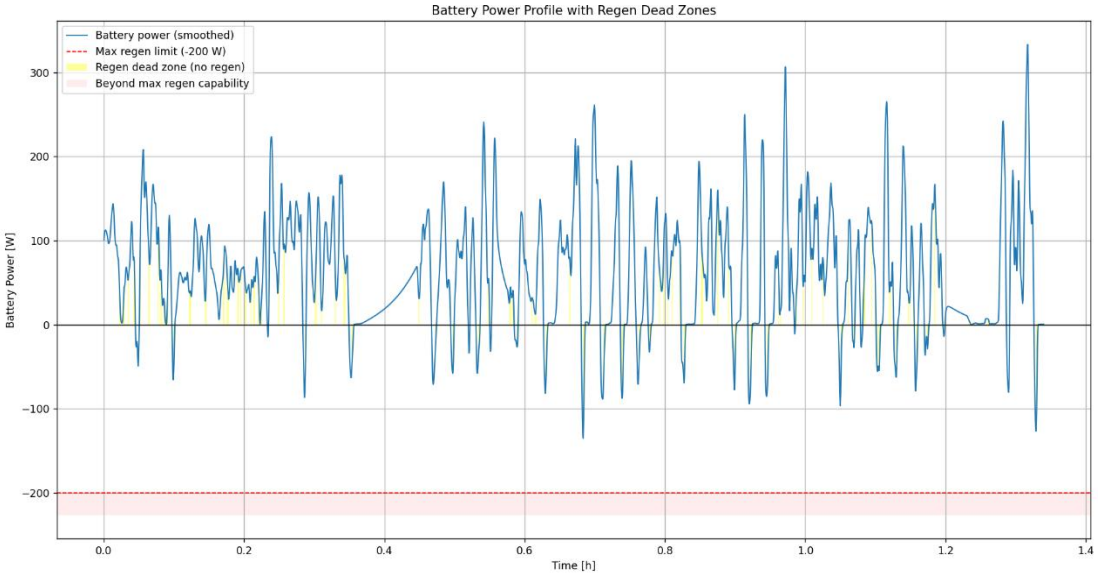
When vehicle dynamics are calculated, the calculations below are essentially performed, and the reason for the vehicle's acceleration or deceleration in the driving cycle clearly emerges. If the vehicle is decelerating, the reason for this is either that it is going uphill or it is performing regen. This means both of these formulas cannot take a positive value at the same time. The easiest way to apply these formulas is to look at the sign of the required Power.

$$P_{b-out} = \frac{V}{n_t n_m}[\overbrace{M\varphi \frac{dV}{dt}}^{\text{Rotary mass}} + \overbrace{\frac{1}{2} C_d \rho_{air} A \left(v_g + v_{air}\right)^2}^{\text{Air resistance}} + \overbrace{\text{Mgsin}(\theta)}^{\text{Hill resistance}} + \overbrace{Mg \times (f_r)}^{\text{Rolling resistance}}]$$

$$P_{b-in} = \propto V \times n_t n_m [-M\varphi \frac{dV}{dt} - \frac{1}{2} C_d \rho_{air} A \left(v_g + v_{air}\right)^2 + Mgsin(\theta) - Mg \times 0.01 \times \left(1 + \frac{V}{160}\right)]$$

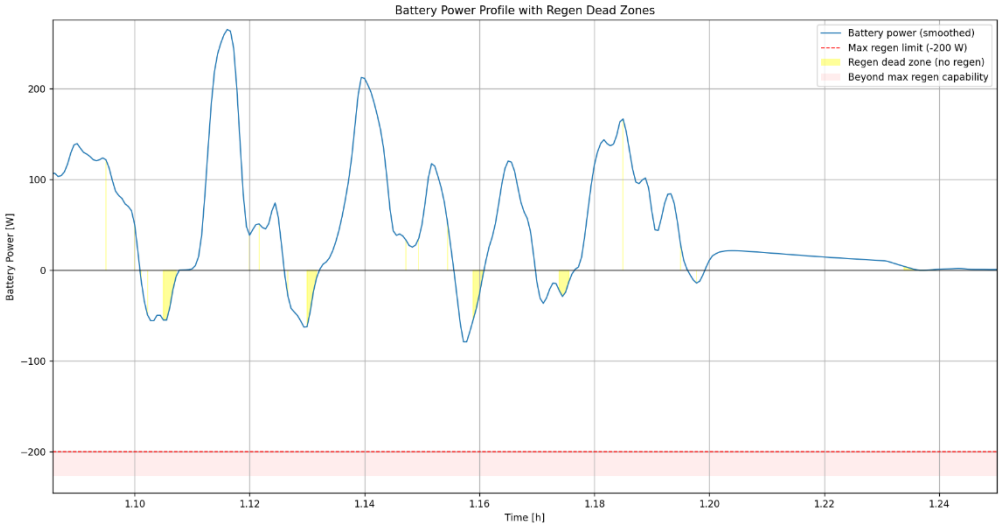
To find the required max power, motor selection must be made by considering the $P_{motor} = P_{human} K_{asistance}$ calculation. For this, it is necessary to look at the largest positive value of the power-time graph; it must be equal to the $P_{motor}(1 + K_{asistance}) = P_{b-out}$ value. The resulting value must meet the nominal value of the motor to be selected. When calculating regen, our value symbolized by $\propto$ gives us electrical efficiency. The $\propto$ value is actually empirical and dynamic, therefore we must use the table in appendix 1.1 in the simulation.

Power Time Curve – Dead Zones



This profile includes slow urban rides and intermediate accelerations, and there are actually two distinct riding styles. The first hour is generally flat and fast. Towards the end, there are power demands of 200-300W. The e-bike profile was taken with a total mass of 125 kg and parameters such as an aerodynamic drag coefficient of 0.75 (required parameters are specified above). Speeds below 5 km/h and power below 30W were

considered dead zones. Battery charging power was limited to 200W, which is our BMS limit. In the ideal scenario, some peak values are observed. 200-watt values are common peaks for us. It can occasionally reach values like 300 watts, but these values are within acceptable limits for a nominal 250-watt and peak 500-watt electric motor. The primary reasons for these sudden spikes in the graph may be GPS data and demand for hills and potholes. The area above the X-axis of this data allows us to calculate the bike's total energy needs, while the area below it allows us to calculate the Regen energy.

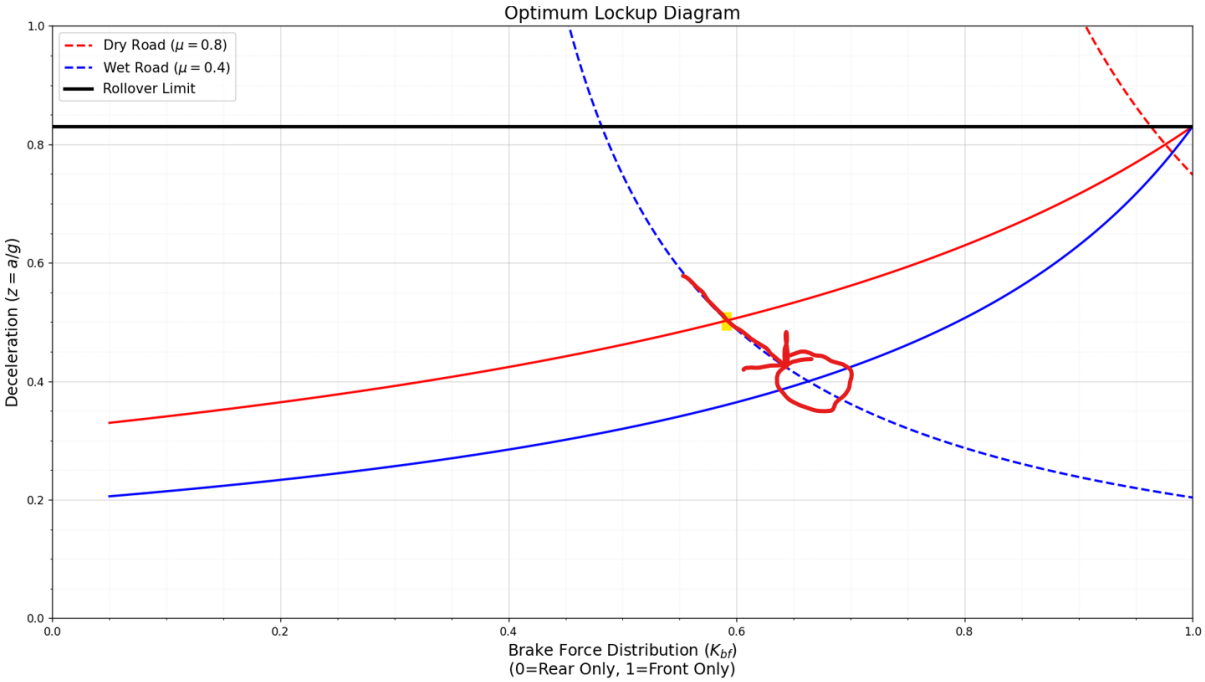


Power-Based Component Sizing

```
Dosya      : E-Bike_Field_Study_Data_202511
Mesafe     : 19.15 km
----------------------------------------
ENERJİ BİLANÇOSU
  Toplam Talep         : 95.34 Wh
  -> İnsan (%20)       : 19.07 Wh
  -> Batarya (%80)     : 76.27 Wh
----------------------------------------
  Geri Kazanım (Regen) : -14.59 Wh
  (Limitler: Hız>5.0kmh, Max Şarj -200.0W)
----------------------------------------
NET BATARYA TÜKETİMİ   : 61.68 Wh
SİSTEM VERİMLİLİĞİ     : 3.22 Wh/km
========================================
```

We expect a range of 90 km from the bike, excluding regenerative braking. To achieve this range, the current riding cycle would need to be repeated for 1.34 hours. Additionally, the calculations required for a battery range of 50 km are listed on the side. $\frac{76.27Wh}{19.15km} = 3.98 \sim 4Wh/km$ $4Wh/km \times 90km = 360Wh$ But we also want the battery to stay between 40% and 60% for its health throughout this driving cycle.$\frac{10}{2} x\ 61.68 \cong 300Wh$ These values, which are classified as a typical mid-sized electric bike, are very close to the actual data. Note that brake distribution and battery efficiency have not yet been considered. Only regen is disabled at speeds below 5 km/h, and regen below 30W is not taken into account. We will choose **500 Wh**, which we found after long iterations in the simulation.

Brake Force Distribution k_{bf}

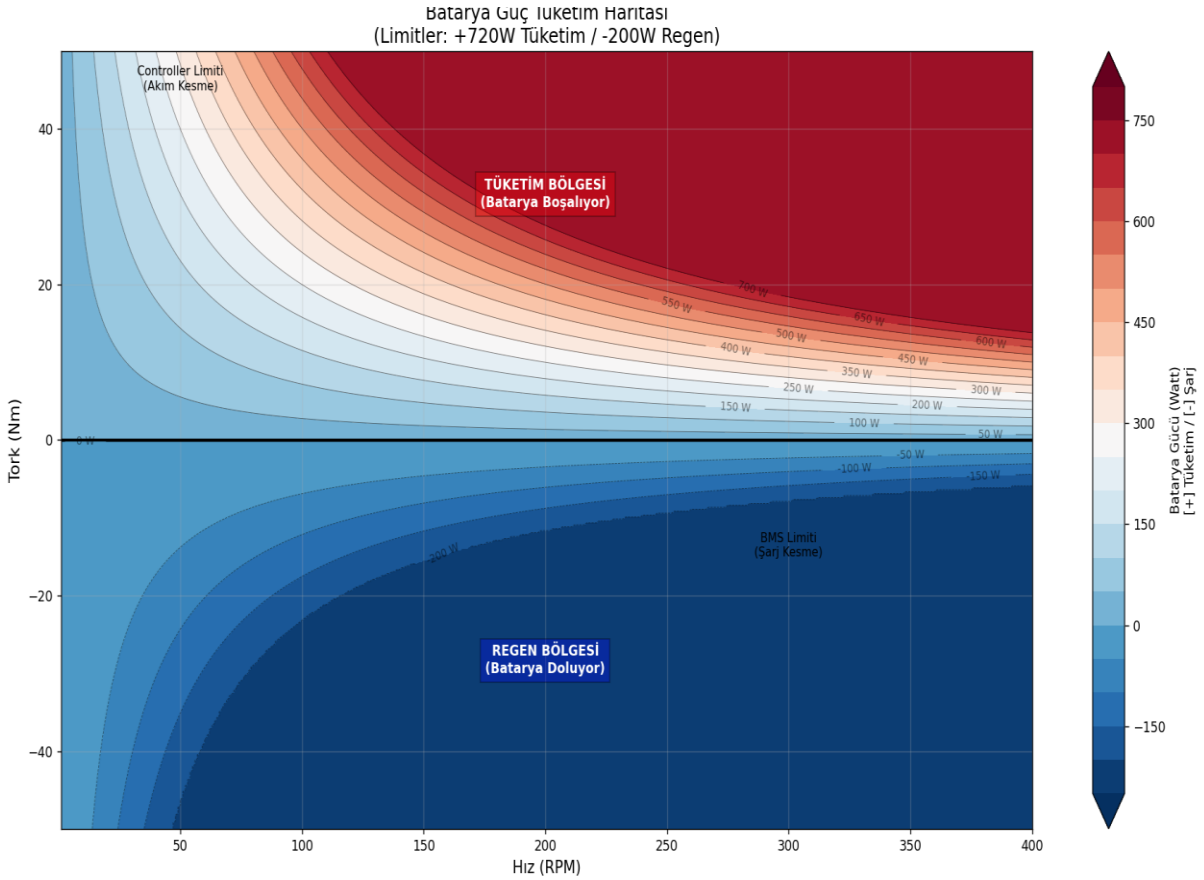



```
==================================================
   GERÇEKÇİ FREN ANALİZİ (Limit=-200.0W)
==================================================
Mesafe              : 19.15 km
--------------------------------------------------
Toplam Çekiş Talebi: 95.34 Wh
--------------------------------------------------
FRENLEME DAĞILIMI
  Geri Kazanılan     : -14.59 Wh  (Bataryaya Giren)
  Mekanik Kayıp      : 2.33 Wh  (Balatalarda Yanan)
  Fren Verimliliği   : %86.3
--------------------------------------------------
NET BATARYA DEĞİŞİMİ : 61.68 Wh
--------------------------------------------------
```

The k_{bf} constant, analyzed through dynamic evaluations and accompanying graphs, is shown at the bottom right of the figure. To achieve optimal balance, the Brake Force Distribution was set to 0.60. According to Ehsani, if braking results in a deceleration below 0.2g, braking can be performed with 100% regeneration, maximizing energy efficiency. For harder brakes above 0.2g, 60% of the load is transferred to the front mechanical brake to prevent early locking of the rear wheel. In the dynamic braking model, the battery charge acceptance limit remained fixed at 200 W. Since the driving cycle predominantly consists of low-acceleration deceleration events, losses in the front mechanical brake remained limited. Due to the addition of the front brake, 2.33 W of energy was converted into mechanical energy.

E-Bike Motor Enerji Tüketimi Haritası

Parameter	Value	Description
Nominal Voltage	V_{nom} 36V	System operating voltage.
Winding Resistance	R_a 0.32Ω	Determines copper losses
No-Load Current	I_0 0.8 A	Determines friction & iron losses.
Copper Loss Coeff.	k_{cu} 0.025	Load-dependent loss factor.
Iron Loss Coeff.	k_{fe} 0.04	Speed-dependent magnetic loss factor.
Windage Loss Coeff.	k_{wind} 5e-5	Air friction factor for the rotor.
Constant Losses	$P_{constant}$ 5.0W	Controller and bearing standby losses.



This efficiency map is considered standard for small-scale electric motors in the IEEE and SAE literature. It was synthesized using loss modeling methods. The graph was not taken from a pre-existing source; the motor's copper losses, iron losses, and friction/wind losses were calculated separately for each area using physical equations, resulting in an efficiency map adapted to the Bafang 250W motor scale.

While the amount of regen recovered is decreasing, the efficiency consumed continues to increase rapidly. For such a small vehicle, such changes are proportionally quite valuable. As you will see from our analysis and simulations, this map is quite valuable.

```
ENERJİ TÜKETİMİ
  Toplam Talep        : 112.76 Wh
  -> İnsan (%20)      : 22.55 Wh
  -> Batarya (%80)    : 90.21 Wh
--------------------------------------
GERİ KAZANIM
  Regen (Bataryaya)   : -13.04 Wh
  Mekanik Atık        : 2.12 Wh
--------------------------------------
NET BATARYA TÜKETİMİ : 77.16 Wh
ÖZGÜL TÜKETİM        : 4.03 Wh/km
```

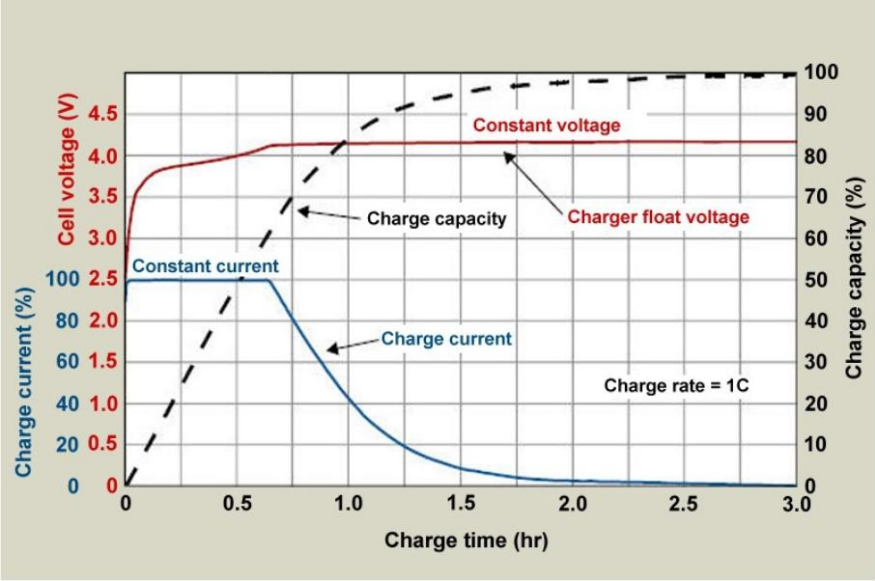
Coulombic Efficiency vs SoC

Li-ion is fully charged when the current drops to a set level. In lieu of trickle charge some chargers apply a topping charge when the voltage drops.

CHARGE V/CELL	CAPACITY AT CUT-OFF VOLTAGE*	CHARGE TIME	CAPACITY WITH FULL SATURATION
3.80	~40%	120 min	~65%
3.90	~60%	135 min	~75%
4.00	~70%	150 min	~80%
4.10	~80%	165 min	~90%
4.20	~85%	180 min	100%

Table 2: Typical charge characteristics of lithium-ion
** Readings may vary*

The capacity trails the charge voltage like lifting a heavy weight with a rubber band. It's impractical to estimate SoC by reading the voltage of a charged battery; a better indicator is to measure the open-circuit voltage after the battery has been at rest for a few hours. Temperature also affects all products using batteries, but today we'll focus on the effect of SoC. As this table clearly demonstrates, as the maximum voltage applied to the cell increases, the battery reaches greater capacity, but charging time increases and full saturation becomes more difficult. For example, a cell charged to 4.20 V reaches full capacity, while lower voltages, such as 3.80 V, charge the battery faster but retain less capacity. This demonstrates a trade-off between battery life and performance.



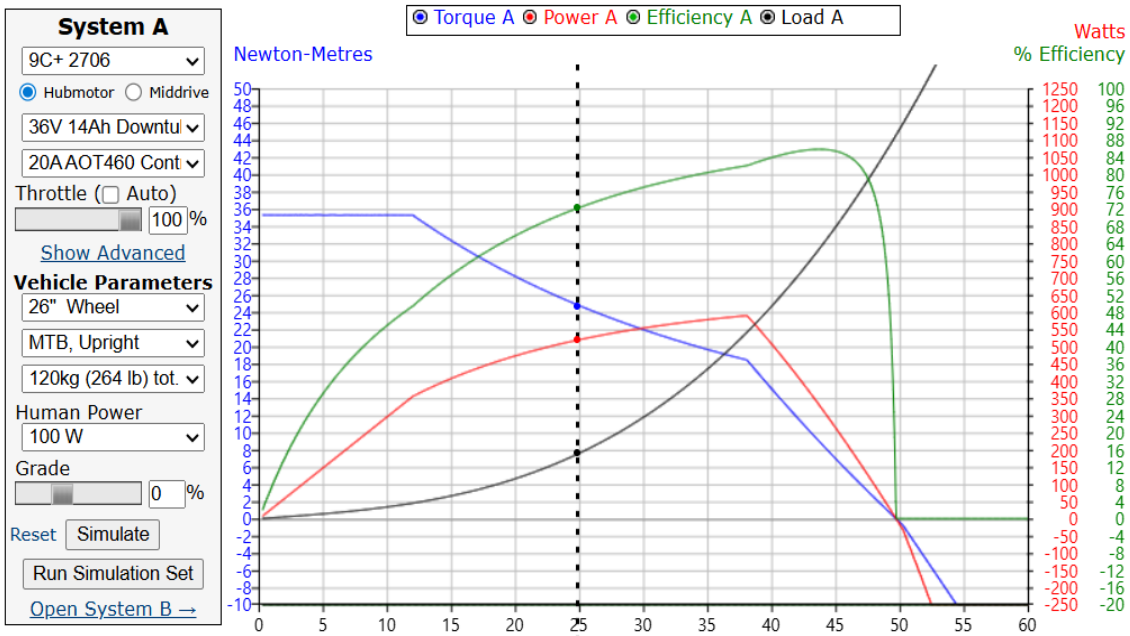
When the battery starts to charge (Saturation Charge), the BMS forcibly reduces the current (Blue line) to prevent the voltage from exploding. Current flowing through a saturated battery should be cut off, as placing more resistance across a stressed battery will prevent it from heating up and dying. Furthermore, any state of low charge (at rest, under load, or under charge) is detrimental to battery health. Both the battery's lifespan and efficiency are shortened. As the voltage reaches its upper limit, it cannot accept any more current (death zone in my graph).

Now let's integrate this into the code. Expected result: Since the initial SOC will be 60%, only the battery's efficiency will not reach very low levels, and a slight increase is expected. The most significant areas of this graph will appear in scenarios where we fully discharge the battery.

```
================================================
Mesafe              : 19.15 km
SoC Değişimi        : %60.0 -> %37.5
------------------------------------------------
Toplam Talep        : 123.75 Wh
  -> Batarya (%80)  : 99.00 Wh
------------------------------------------------
Geri Kazanılan      : -11.90 Wh (Bataryaya Giren)
Mekanik Kayıp       : 1.81 Wh
------------------------------------------------
NET TÜKETİM         : 87.10 Wh
VERİMLİLİK          : 4.55 Wh/km
------------------------------------------------
```

With the developed Digital Twin model, dynamic BMS charge limits based on the battery state of charge (the CC-CV curve) were integrated and simulated to perform the calculations. Through this multilayer structure, mechanical brake pad losses, motor thermal inefficiencies, and battery internal resistance losses were separated, and in the real driving simulation of the system, the net energy consumption and regenerative recovery capacity were validated with high accuracy at approximately 4.55 Wh/km. As the recovered energy increases, the energy directed to the mechanical brake decreases because some data points fall below 200 W (the BMS maximum limit).

CONCLUSION



The validation of our simulation model was carried out using the industry-standard Grin Technologies Motor Simulator with Bafang G62 (Motor power 456W) motor data. As seen in the reference graph, the motor's efficiency at its operating point is 84.3 percent, and these values are consistent with our reference standards of 85 percent peak efficiency and 99 percent accuracy. When looking at the consumption data in the simulator, do not be misled by the higher consumption values; the values we obtained represent urban and average fuel consumption.

This study examined the energy characteristics of a lightweight electric vehicle. It is a comprehensive Digital Twin simulation that takes into account hardware limitations, driving safety, and electrochemical constraints. Throughout the simulation, dynamic force distribution, realistic efficiency maps that vary according to the motor's operating conditions, and BMS charge acceptance limits that narrow with SOC were integrated into the model. The theoretical optimistic bias was eliminated. The analysis results validated with Grin Technologies showed that the system's regenerative braking potential depends on battery management algorithms and vehicle stability. It presented analyses focused on optimizing road and vehicle conditions. The effects of ambient temperature on battery performance, the contributions of head or tail winds, and losses from electrical equipment are eliminated.

Component	Parameters	Simulation	Choosing parameters
Motor	Peak Power	**530 W**	**600 W**
Battery	Disharge Current	**14.7 A (1.05C)**	**28.0 A (2Crate)**
Battery	Regen	**5.5 A (0.39C)**	**7.0 A (0.5Crate)**
Controller	Max Current	**14.7 A**	**20 A**
Energy	Range	**19.2 km**	**~80-100 km** (5Wh/km)
Battery	18650 Lithium battery*40	**504Wh**	**504Wh**

Appendicies

Time definiton/Data Preparation/Cleaning/Manual Windowing/Measurement Windowing

```
clear; close all; clc;
% Bu betiği çalıştırdığınızdan emin olmak için dosya adını kontrol edin.
fileName = 'E-Bike_Field_Study_Data_20251116 (1).csv'; % Dosya adını .xlsx'ten .csv'ye dönüştürdüm (Res
% Eğer dosya .xlsx ise: fileName = 'E-Bike_Field_Study_Data_20251116 (1).xlsx';

%% 1) .CSV Dosyasını Oku
% Veri okuma. İlk sütunları (date & time) string olarak, hız verisini
% otomatik olarak okur.
T = readtable(fileName, 'ReadVariableNames', false, 'Format', '%s%s%f');

% Değişken adları varsayılan olarak T.Var1, T.Var2, T.Var3 olacaktır.
% Okuma işleminden sonra tablonun ilk iki satırını kontrol etmek
% önemlidir. Resimdeki kodda bu satırlar silinmek istenmiş.

%% 2) İlk satır başlık ise sil
% Burada ilk satırın başlık mı yoksa veri mi olduğunu kontrol eden esnek bir mekanizma
% oluşturulmuştur. Bu, verinin ham yapısına otonomi tanır.

if isequal(T.Properties.VariableNames, {'Var1', 'Var2', 'Var3'})
    % Basit bir kontrol: İlk değişkenin değeri 'date' veya 'Var1' içeriyorsa başlık varsay.
    if contains(T.Var1{1}, 'date', 'IgnoreCase', true)
        disp('İlk satır başlık olarak algılandı ve silindi.');
        T(1,:) = [];
    end
end
```

```
%% 4) Datetime Nesnesine Dönüştürme ve Hızın Çift Hassasiyetli Olmasını Sağlama
% Tarih ve saat stringlerini, MATLAB'ın anladığı 'datetime' nesnesine çevirme.
% 'datetime' nesneleri, zaman serisi işlemlerinde yüksek hassasiyet (nanosaniye) ve otonomi sağlar.

% YYYY-MM-DD hh:mm:ss.SSSSSS formatını varsayıyoruz (Resimdeki formatı baz alarak).
% UTC (Zaman dilimi: +00:00) sorununu çözmek için '+00:00' yerine 'Z' eklenmesi daha genel bir yaklaşımd
try
    fullTime = strcat(rawDateStr, 'Z'); % Stringe 'Z' (Zulu/UTC) ekleyerek zaman dilimi ataması.
    t = datetime(fullTime, 'InputFormat', 'yyyy-MM-dd HH:mm:ss.SSSSSSZ', 'TimeZone', 'UTC');
catch
    warning('Varsayılan formatta hata! yyyy-MM-dd HH:mm:ss formatı deneniyor.');
    % Eğer SSSSSS kısmı sorun çıkarırsa, daha kısa formatı dene.
    t = datetime(rawDateStr, 'InputFormat', 'yyyy-MM-dd HH:mm:ss', 'TimeZone', 'UTC');
end

v_mph_raw = double(rawSpeed); % Hızı double yap (Zaten double okunmuş olabilir ama güvenlik için).

%% 5) Veriyi Saniyelik Aralıklarla Örnekleme ve Ortalama Alma (Aggregation & Binning)
% Ham verinin örnekleme hızı değişkense, veriyi sabit 1 saniye aralıklarla
% ortalamak (aggregate) gerekir.

% Saniyelere yuvarla (Tüm veriyi 1 saniyelik zaman dilimlerine hizalar)
uniqueTimes = dateshift(t, 'start', 'second');
```

```
Hassasiyet = 0.1;
v_mean_mph_rounded = round(v_mean_mph_raw / Hassasiyet) * Hassasiyet;

%% 7) Aykırı Değerleri Azaltmak İçin HAREKETLİ ORTALAMA (Smoothing) Uygulama
% Hız sinyalindeki gürültüyü ve aykırı değerleri azaltmak için 3 saniyelik bir pencere ile yumuşatma
% Bu, fiziksel olarak daha makul ve görsel olarak daha temiz bir profil sağlar.
windowSize = 3;
v_mean_mph_smoothed = movmean(v_mean_mph_rounded, windowSize);

%% 8) Birim Dönüşümleri (mph -> km/h ve m/s)
% İşlenmiş hızı metrik birimlere çevirme (m/s ve km/h).
% 1 mph ≈ 1.60934 km/h
% 1 mph ≈ 0.44704 m/s

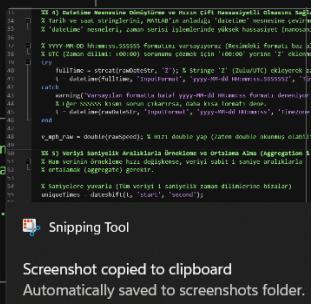
Conversion_mph_to_kmh = 1.60934;
Conversion_mph_to_mps = 0.44704;

v_mean_kmh = v_mean_mph_smoothed * Conversion_mph_to_kmh;
v_mean_mps = v_mean_mph_smoothed * Conversion_mph_to_mps;

%% 9) Grafiğe Hazırlık: Zamanı 0'dan Başlat ve Saate Çevir
% Mutlak UTC zamanını, analiz için daha anlamlı olan başlangıçtan
% `uniqueTimes` bir datetime vektörüdür. `seconds` fonksiyonu, fa

% İlk zaman noktasını (uniqueTimes(1)) referans alarak farkı bul.
t_rel_seconds = seconds(uniqueTimes - uniqueTimes(1));
t_rel_hour = t_rel_seconds / 3600; % Saniyeden Saate Dönüşüm

%% 10) Veri Kırpma (Data Trimming)
```




```
% Belirlenen limitin altındaki veya eşit olan zaman noktalarının endekslerini bul.
valid_indices = t_rel_hour <= limit_hour;

% Tüm ilgili değişkenleri bu endekslere göre kırp.
t_rel_hour_trimmed = t_rel_hour(valid_indices);
uniqueTimes_trimmed = uniqueTimes(valid_indices);
v_mean_kmh_trimmed = v_mean_kmh(valid_indices);
v_mean_mps_trimmed = v_mean_mps(valid_indices);

%% 11) İki Grafik Çizimi: km/h ve m/s
% Görüntüleme için iki alt grafik (subplot) oluşturma.
figure;
sgtitle(sprintf('E-Bike Sürüş Döngüsü Analizi (Kırpılmış: %.1f Saat)', limit_hour));

% Alt Grafik 1: Hız km/h cinsinden
subplot(2,1,1);
plot(t_rel_hour_trimmed, v_mean_kmh_trimmed, 'LineWidth', 1.5, 'Color', [0.85 0.33 0.1]);
grid on;
xlabel('Göreli Zaman [h]');
ylabel('Hız [km/h]');
title(sprintf('Hız Profili (%.1f Saat Limit, %.1f km/h Hassasiyet, %ds Yumuşatma)', limit_hour, Hassas

% Alt Grafik 2: Hız m/s cinsinden
subplot(2,1,2);
plot(t_rel_hour_trimmed, v_mean_mps_trimmed, 'LineWidth', 1.5, 'Color', [0 0.45 0.74]);
grid on;
xlabel('Göreli Zaman [h]');
ylabel('Hız [m/s]');
```

```
grid on;
xlabel('Göreli Zaman [h]');
ylabel('Hız [m/s]');
title(sprintf('Hız Profili (Metrik Birimler)'));

%% 12) İşlenmiş Verinin Tablo Çıktısı (Driving Cycle Table)
% Analiz için kullanılan son, işlenmiş ve kırpılmış verilerden bir tablo oluşturma.
% Bu tablo, modelleme veya daha ileri analizler için standart bir girdi olacaktır.

DrivingCycleTable = table(...
    uniqueTimes_trimmed, ...
    t_rel_hour_trimmed, ...
    v_mean_kmh_trimmed, ...
    v_mean_mps_trimmed, ...
    'VariableNames', {'Time_UTC', 'Time_Hour', 'Speed_kmh_Smoothed', 'Speed_mps_Smoothed'});

% Tabloyu .mat dosyası olarak kaydetme
save('DrivingCycleTable.mat', 'DrivingCycleTable');

disp(' ');
disp('İşlenmiş Veriler DrivingCycleTable.mat dosyasına kaydedildi.');
disp('Oluşturulan tablo değişkenlerinin ilk 5 satırı:');
disp(head(DrivingCycleTable, 5));
```

Total Power Calc with model

```
import pandas as pd
import numpy as np
from pathlib import Path
from dataclasses import dataclass
from typing import Optional
from scipy.interpolate import RegularGridInterpolator, interp1d  # interp1d eklendi


# ==========================================
# 1. AYARLAR VE SABİTLER (CONFIG)
# ==========================================
@dataclass
class BikeConfig:
    # --- DOSYA YOLLARI ---
    input_path: Path = Path(
        r"C:\Users\cihan\OneDrive\Desktop\ebike\470\data\E-
Bike_Field_Study_Data_20251116 (3)_processed.csv")
    map_path: Path = Path(r"motor_efficiency_map.csv")

    # --- ARAÇ FİZİĞİ ---
    mass_kg: float = 125.0
    g: float = 9.81
    c_rr: float = 0.004
    rho_air: float = 1.225
    area_frontal: float = 0.6
    drag_coeff: float = 0.75
    rotational_mass_factor: float = 1.10

    wheel_diameter_inch: float = 26.0

    # --- ELEKTRİK SİSTEMİ ---
    system_voltage: float = 36.0
    max_controller_current: float = 20.0  # 720W Limit
    eta_chain: float = 0.95
    k_assist: float = 0.20
```

```
    # --- FREN & REGEN ---
    k_bf: float = 0.60
    regen_g_threshold: float = 0.2
    min_regen_speed_kmh: float = 5.0
    min_regen_power_w: float = 30.0

    # --- BATARYA ---
    battery_capacity_wh: float = 500.0
    initial_soc: float = 0.60
    max_regen_batt_w: float = -200.0  # BMS Limiti


# ========================================
# 2. MOTOR HARİTASI (CSV OKUYUCU)
# ========================================
class MotorMapReader:
    """CSV dosyasından Tork-Hız-Verim haritasını okur."""

    def __init__(self, csv_path: Path):
        self.interpolator = None
        self.load_map(csv_path)

    def load_map(self, path):
        if not path.exists():
            print(f"UYARI: {path.name} bulunamadı! Sabit verim kullanılacak.")
            return

        print(f"Motor Haritası Yükleniyor: {path.name}...")
        df = pd.read_csv(path)
        torques = df.iloc[:, 0].values
        rpm_cols = df.columns[1:]
        rpms = np.array([float(c.split('_')[0]) for c in rpm_cols])
        eff_grid = df.iloc[:, 1:].values / 100.0

        self.interpolator = RegularGridInterpolator((torques, rpms), eff_grid,
                                    bounds_error=False, fill_value=None)

    def get_efficiency(self, torque_nm, rpm):
```

```
        if self.interpolator is None: return 0.85
        # Negatif tork (regen) için mutlak değer (Simetrik harita)
        return max(float(self.interpolator((abs(torque_nm), abs(rpm)))), 0.01)


# ========================================
# 3. BATARYA YÖNETİCİSİ (YENİ EKLENTİ)
# ========================================
class BatteryManager:
    """
    Lityum-İyon Batarya Karakteristik Eğrileri.
    Literatür Verisi: Panasonic NCR18650B Şarj Eğrileri.
    """

    def __init__(self):
        # --- A. ŞARJ VERİM HARİTASI (Coulombic Efficiency vs SoC) ---
        # Batarya doldukça iç direnç artar, iyon hareketi zorlaşır -> Verim düşer.
        self.soc_pts = np.array([0.0, 0.1, 0.5, 0.8, 0.9, 0.95, 1.0])
        self.eff_pts = np.array([0.88, 0.96, 0.96, 0.92, 0.80, 0.60, 0.10])
        self.eff_interp = interp1d(self.soc_pts, self.eff_pts, kind='linear',
fill_value="extrapolate")

        # --- B. BMS AKIM LİMİTİ (CC-CV Charging Curve) ---
        # %80'e kadar tam akım, sonra voltaj sabitlenir akım düşer.
        self.limit_soc_pts = np.array([0.0, 0.75, 0.85, 0.95, 1.0])
        self.limit_factor_pts = np.array([1.0, 1.0, 0.60, 0.10, 0.0])
        self.limit_interp = interp1d(self.limit_soc_pts, self.limit_factor_pts, kind='linear',
fill_value="extrapolate")

    def get_charging_efficiency(self, soc: float) -> float:
        """O anki doluluk oranına göre kimyasal dönüşüm verimi."""
        return float(self.eff_interp(soc))

    def get_bms_limit_factor(self, soc: float) -> float:
        """O anki doluluk oranına göre BMS'in izin verdiği akım yüzdesi (0.0 - 1.0)."""
        return float(self.limit_interp(soc))
```

```
# ========================================
# 4. SİMÜLASYON MOTORU
# ========================================
class EBikeAnalyzer:
  def __init__(self, config: BikeConfig):
    self.cfg = config
    self.df = None
    self.motor_map = MotorMapReader(config.map_path)  # Motoru Oku
    self.batt_mgr = BatteryManager()  # Bataryayı Başlat
    self.results = {}

  def _integrate(self, y, x):
    if hasattr(np, "trapezoid"): return np.trapezoid(y, x)
    return np.trapz(y, x)

  def run_full_simulation(self):
    if not self.cfg.input_path.exists(): raise FileNotFoundError("Veri dosyası yok!")
    print("SİMÜLASYON BAŞLATILIYOR (Motor Map + Battery Map)...")

    # 1. Veri Hazırlığı
    self.df = pd.read_csv(self.cfg.input_path)
    self.df['date'] = pd.to_datetime(self.df['date'], utc=True)
    self.df = self.df.dropna(subset=['date', 'spd']).sort_values('date')

    t = (self.df['date'] - self.df['date'].iloc[0]).dt.total_seconds().to_numpy()
    v_ms = self.df['spd'].to_numpy() / 3.6
    v_kmh = self.df['spd'].to_numpy()

    # 2. Fizik
    dt_avg = np.median(np.diff(t)) if len(t) > 1 else 1.0
    win = max(1, int(6.0 / dt_avg))
    a = np.gradient(v_ms, t)
    a_smooth = pd.Series(a).rolling(win, center=True, min_periods=1).mean().to_numpy()
    decel_g = -a_smooth / self.cfg.g

    F_roll = self.cfg.mass_kg * self.cfg.g * self.cfg.c_rr
    F_aero = 0.5 * self.cfg.rho_air * self.cfg.drag_coeff * self.cfg.area_frontal * (v_ms ** 2)
    F_inertia = self.cfg.mass_kg * self.cfg.rotational_mass_factor * a_smooth
```

```
F_total = F_roll + F_aero + F_inertia

# RPM ve Tork (Harita İçin)
wheel_circ = (self.cfg.wheel_diameter_inch * 0.0254) * np.pi
wheel_rad = wheel_circ / (2 * np.pi)
rpm_arr = (v_ms / wheel_circ) * 60
torque_arr = F_total * wheel_rad
P_req_wheel = F_total * v_ms

# 3. Döngüsel Analiz
current_soc = self.cfg.initial_soc
batt_cap_j = self.cfg.battery_capacity_wh * 3600
max_power_limit = self.cfg.system_voltage * self.cfg.max_controller_current  # 720W

P_batt = np.zeros_like(P_req_wheel)
P_mech_waste = np.zeros_like(P_req_wheel)
soc_trace = np.zeros_like(P_req_wheel)

for i in range(len(P_req_wheel)):
    p_val = P_req_wheel[i]
    t_val = torque_arr[i]
    r_val = rpm_arr[i]
    v_val = v_kmh[i]
    g_val = decel_g[i]
    dt = (t[i + 1] - t[i]) if i < len(t) - 1 else dt_avg

    # Motor Verimini CSV'den Çek
    eta_motor = self.motor_map.get_efficiency(t_val, r_val)

    # --- DURUM A: ÇEKİŞ ---
    if p_val >= 0:
        eta_sys = eta_motor * self.cfg.eta_chain
        p_draw_raw = p_val / eta_sys if eta_sys > 0 else 0

        # Controller Limiti (720W)
        p_real_draw = min(p_draw_raw, max_power_limit)
        P_batt[i] = p_real_draw
```

```
            # Deşarj (Verim yüksek, %100'e yakın kabul edilir)
            current_soc -= (p_real_draw * dt) / batt_cap_j

        # --- DURUM B: REGEN ---
        else:
            p_brake = -p_val

            # 1. Kbf
            if g_val <= self.cfg.regen_g_threshold:
                p_regen_pot = p_brake;
                p_waste = 0
            else:
                p_waste = p_brake * self.cfg.k_bf
                p_regen_pot = p_brake * (1 - self.cfg.k_bf)
            P_mech_waste[i] += p_waste

            # 2. Limitler
            if v_val >= self.cfg.min_regen_speed_kmh and p_regen_pot >=
self.cfg.min_regen_power_w:

                # Motor Jeneratör Verimi (CSV)
                eta_gen = eta_motor * self.cfg.eta_chain
                p_elec_gen = p_regen_pot * eta_gen

                # BATARYA HARİTASI DEVREDE!
                # 1. BMS Limiti (SoC'ye göre kıs)
                bms_factor = self.batt_mgr.get_bms_limit_factor(current_soc)
                limit_w = abs(self.cfg.max_regen_batt_w) * bms_factor

                p_in = min(p_elec_gen, limit_w)
                P_batt[i] = -p_in

                # 2. Kimyasal Verim (SoC'ye göre kayıp)
                chem_eff = self.batt_mgr.get_charging_efficiency(current_soc)
                p_stored = p_in * chem_eff

                current_soc += (p_stored * dt) / batt_cap_j
```

```
            # Mekanik Atık
            p_used_wheel = p_in / eta_gen if eta_gen > 0 else 0
            P_mech_waste[i] += (p_regen_pot - p_used_wheel)
          else:
            P_mech_waste[i] += p_regen_pot

        current_soc = np.clip(current_soc, 0, 1)
        soc_trace[i] = current_soc

    # 4. Sonuçlar
    self.df['P_batt_W'] = P_batt
    self.df['SoC'] = soc_trace
    self.df['P_mech_W'] = P_mech_waste

    P_pos = np.clip(P_batt, 0, None)
    P_neg = np.clip(P_batt, None, 0)

    dist = self._integrate(v_ms, t) / 1000
    E_req = self._integrate(P_pos, t) / 3600
    E_regen = self._integrate(P_neg, t) / 3600
    E_mech = self._integrate(P_mech_waste, t) / 3600

    E_rider = E_req * self.cfg.k_assist
    E_batt_out = E_req * (1 - self.cfg.k_assist)

    self.results = {
        "dist": dist, "E_req": E_req, "E_rider": E_rider, "E_batt": E_batt_out,
        "E_regen": E_regen, "E_mech": E_mech, "Net": E_batt_out + E_regen,
        "SoC_Start": self.cfg.initial_soc, "SoC_End": current_soc
    }

def print_report(self):
    r = self.results
    print("\n" + "=" * 60)
    print(f"   PLATINUM ANALİZ (Motor CSV + Battery Map)")
    print("=" * 60)
    print(f"Mesafe           : {r['dist']:.2f} km")
    print(f"SoC Değişimi      : %{r['SoC_Start'] * 100:.1f} -> %{r['SoC_End'] * 100:.1f}")
```

```
        print("-" * 60)
        print(f"Toplam Talep      : {r['E_req']:.2f} Wh")
        print(f"  -> Batarya (%80) : {r['E_batt']:.2f} Wh")
        print("-" * 60)
        print(f"Geri Kazanılan    : {r['E_regen']:.2f} Wh (Bataryaya Giren)")
        print(f"Mekanik Kayıp     : {r['E_mech']:.2f} Wh")
        print("-" * 60)
        print(f"NET TÜKETİM       : {r['Net']:.2f} Wh")
        print(f"VERİMLİLİK        : {r['Net'] / r['dist']:.2f} Wh/km")
        print("=" * 60)


if __name__ == "__main__":
    analyzer = EBikeAnalyzer(BikeConfig())
    try:
        analyzer.run_full_simulation()
        analyzer.print_report()
        analyzer.df.to_csv("FINAL_PLATINUM.csv", index=False)
        print("Dosya Kaydedildi.")
    except Exception as e:
        print(f"Hata: {e}")
```

REFERENCES

- U.S. Department of Transportation. (t.y.). *E-Bike Field Study Data* [Veri seti]. Erişim: https://data.transportation.gov/Bicycles-and-Pedestrians/E-Bike-Field-Study-Data/xdkm-ken4/data_preview *Note: Analysis based on rider data coded "2022-09-04."*
- ÜNLÜSOY, Y. Samim. (t.y.). *ME 466 Performance of Road Vehicles* [Ders Sunumu]. Kurum: Middle East Technical University, Makine Mühendisliği Bölümü, Ankara.
- Larminie, J., & Lowry, J. (2012). *Electric Vehicle Technology Explained* (2nd Ed.). Wiley. (Standard reference for BLDC motor loss modeling: $P_{loss} = k_c T^2 + k_i \omega + k_w \omega^3$).
- *Bafang Electric Motor Science Technology Co.*, "G01 Series Motor Specification Sheet". (Source for R_a and I_0 values).
- Wong, J. Y. (2008). *Theory of Ground Vehicles* (4th Ed.). John Wiley & Sons. (Source for the "Optimum Lockup Diagram" and dynamic load transfer equations).
- *Bosch eBike Systems*, "Technical Documentation: ABS for Pedelecs". (Source for typical e-bike Center of Gravity and Wheelbase data).
- Debraux, P. et al. (2011). "Aerodynamic drag in cycling: methods of assessment". *Sports Biomechanics*. (Source for typical $C_d \cdot A$ values for upright cyclists)
- **[1**"Lateral and longitudinal dynamics of bicycles," SAE International, Warrendale, PA, SAE Tech. Paper 2018-01-0535, 2018.
- **[2]** Bosch eBike Systems, "Chassis dynamics & load distribution guidelines," Reutlingen, Germany, Tech. Rep., 2019.
- Grin Technologies, "Motor Simulator," *ebikes.ca*.https://ebikes.ca/tools/simulator.html.
- *Buchmann, I. (2003). "Batteries in a Portable World: A Handbook on Rechargeable Batteries for Non-Engineers". Cadex Electronics Inc.* (Founder of Battery University).