**TASK:**

Battery Charging and Discharging

* Model Li-on battery charging /discharging behaviour
* **Inputs** : Capacity, Current, Voltage
* **Ouputs: SOC(**state of charge),efficiency
* **Libraries:** numpy, matplotlib
* **Application :** Ev battery Mangment systems

**SOURCE CODE:**

import numpy as np

import matplotlib.pyplot as plt import numpy as np

import matplotlib.pyplot as plt

class LiionBattery:

    def \_\_init\_\_(self, capacity\_Ah, initial\_soc, voltage\_V, charging\_efficiency=0.95, discharging\_efficiency=0.95):

        if not 0 <= initial\_soc <= 1:

            raise ValueError("Initial SOC must be between 0 and 1")

        self.capacity\_Ah = capacity\_Ah

        self.soc = initial\_soc

        self.voltage\_V = voltage\_V

        self.charging\_efficiency = charging\_efficiency

        self.discharging\_efficiency = discharging\_efficiency

    def update\_soc(self, current\_A, time\_step\_h):

        """Updates the battery's state of charge."""

        if current\_A > 0:  # Charging

            charge\_change\_Ah = current\_A \* time\_step\_h \* self.charging\_efficiency

        else:  # Discharging

            charge\_change\_Ah = current\_A \* time\_step\_h / self.discharging\_efficiency

        self.soc += charge\_change\_Ah / self.capacity\_Ah

        self.soc = max(0, min(1, self.soc)) # Ensure SOC stays within [0, 1]

    def get\_current\_voltage(self):

        """Returns the current voltage (simplified model assumes constant voltage)."""

        return self.voltage\_V

    def calculate\_efficiency(self, current\_A):

        """Calculates efficiency based on current (simplified model assumes constant efficiency)."""

        if current\_A > 0: # Charging

            return self.charging\_efficiency

        elif current\_A < 0: # Discharging

            return self.discharging\_efficiency

        else: # No current

            return 1.0

# 1. Define the total simulation time and the time step

total\_time\_h = 10  # Total simulation time in hours

time\_step\_h = 0.1  # Time step in hours

num\_steps = int(total\_time\_h / time\_step\_h)

# 2. Create an instance of the LiionBattery class

battery = LiionBattery(capacity\_Ah=50, initial\_soc=0.5, voltage\_V=350)

# 3. Initialize lists to store simulation results

time\_list = []

soc\_list = []

current\_list = []

voltage\_list = []

efficiency\_list = []

# 4. Loop through the simulation time

for i in range(num\_steps):

    current\_time = i \* time\_step\_h

    # Define the current at the current time step

    # Example: Charging for the first half, discharging for the second half

    if current\_time < total\_time\_h / 2:

        current\_A = 20  # Charging current

    else:

        current\_A = -15  # Discharging current

    # Update the battery's state

    battery.update\_soc(current\_A, time\_step\_h)

    current\_voltage = battery.get\_current\_voltage()

    current\_efficiency = battery.calculate\_efficiency(current\_A)

    # Append results to lists

    time\_list.append(current\_time)

    soc\_list.append(battery.soc)

    current\_list.append(current\_A)

    voltage\_list.append(current\_voltage)

    efficiency\_list.append(current\_efficiency)

# Create a figure and a set of subplots

fig, axes = plt.subplots(2, 1, figsize=(10, 8))

# Plot SOC over time

axes[0].plot(time\_list, soc\_list)

axes[0].set\_xlabel('Time (h)')

axes[0].set\_ylabel('State of Charge (SOC)')

axes[0].set\_title('Battery State of Charge over Time')

# Plot efficiency over time

axes[1].plot(time\_list, efficiency\_list)

axes[1].set\_xlabel('Time (h)')

axes[1].set\_ylabel('Efficiency')

axes[1].set\_title('Battery Efficiency over Time')

# Adjust layout and display plots

plt.tight\_layout()

plt.show()

class LiionBattery:

    def \_\_init\_\_(self, capacity\_Ah, initial\_soc, voltage\_V, charging\_efficiency=0.95, discharging\_efficiency=0.95):

        if not 0 <= initial\_soc <= 1:

            raise ValueError("Initial SOC must be between 0 and 1")

        self.capacity\_Ah = capacity\_Ah

        self.soc = initial\_soc

        self.voltage\_V = voltage\_V

        self.charging\_efficiency = charging\_efficiency

        self.discharging\_efficiency = discharging\_efficiency

    def update\_soc(self, current\_A, time\_step\_h):

        """Updates the battery's state of charge."""

        if current\_A > 0:  # Charging

            charge\_change\_Ah = current\_A \* time\_step\_h \* self.charging\_efficiency

        else:  # Discharging

            charge\_change\_Ah = current\_A \* time\_step\_h / self.discharging\_efficiency

        self.soc += charge\_change\_Ah / self.capacity\_Ah

        self.soc = max(0, min(1, self.soc)) # Ensure SOC stays within [0, 1]

    def get\_current\_voltage(self):

        """Returns the current voltage (simplified model assumes constant voltage)."""

        return self.voltage\_V

    def calculate\_efficiency(self, current\_A):

        """Calculates efficiency based on current (simplified model assumes constant efficiency)."""

        if current\_A > 0: # Charging

            return self.charging\_efficiency

        elif current\_A < 0: # Discharging

            return self.discharging\_efficiency

        else: # No current

            return 1.0

# 1. Define the total simulation time and the time step

total\_time\_h = 10  # Total simulation time in hours

time\_step\_h = 0.1  # Time step in hours

num\_steps = int(total\_time\_h / time\_step\_h)

# 2. Create an instance of the LiionBattery class

battery = LiionBattery(capacity\_Ah=50, initial\_soc=0.5, voltage\_V=350)

# 3. Initialize lists to store simulation results

time\_list = []

soc\_list = []

current\_list = []

voltage\_list = []

efficiency\_list = []

# 4. Loop through the simulation time

for i in range(num\_steps):

    current\_time = i \* time\_step\_h

    # Define the current at the current time step

    # Example: Charging for the first half, discharging for the second half

    if current\_time < total\_time\_h / 2:

        current\_A = 20  # Charging current

    else:

        current\_A = -15  # Discharging current

    # Update the battery's state

    battery.update\_soc(current\_A, time\_step\_h)

    current\_voltage = battery.get\_current\_voltage()

    current\_efficiency = battery.calculate\_efficiency(current\_A)

    # Append results to lists

    time\_list.append(current\_time)

    soc\_list.append(battery.soc)

    current\_list.append(current\_A)

    voltage\_list.append(current\_voltage)

    efficiency\_list.append(current\_efficiency)

# Create a figure and a set of subplots

fig, axes = plt.subplots(2, 1, figsize=(10, 8))

# Plot SOC over time

axes[0].plot(time\_list, soc\_list)

axes[0].set\_xlabel('Time (h)')

axes[0].set\_ylabel('State of Charge (SOC)')

axes[0].set\_title('Battery State of Charge over Time')

# Plot efficiency over time

axes[1].plot(time\_list, efficiency\_list)

axes[1].set\_xlabel('Time (h)')

axes[1].set\_ylabel('Efficiency')

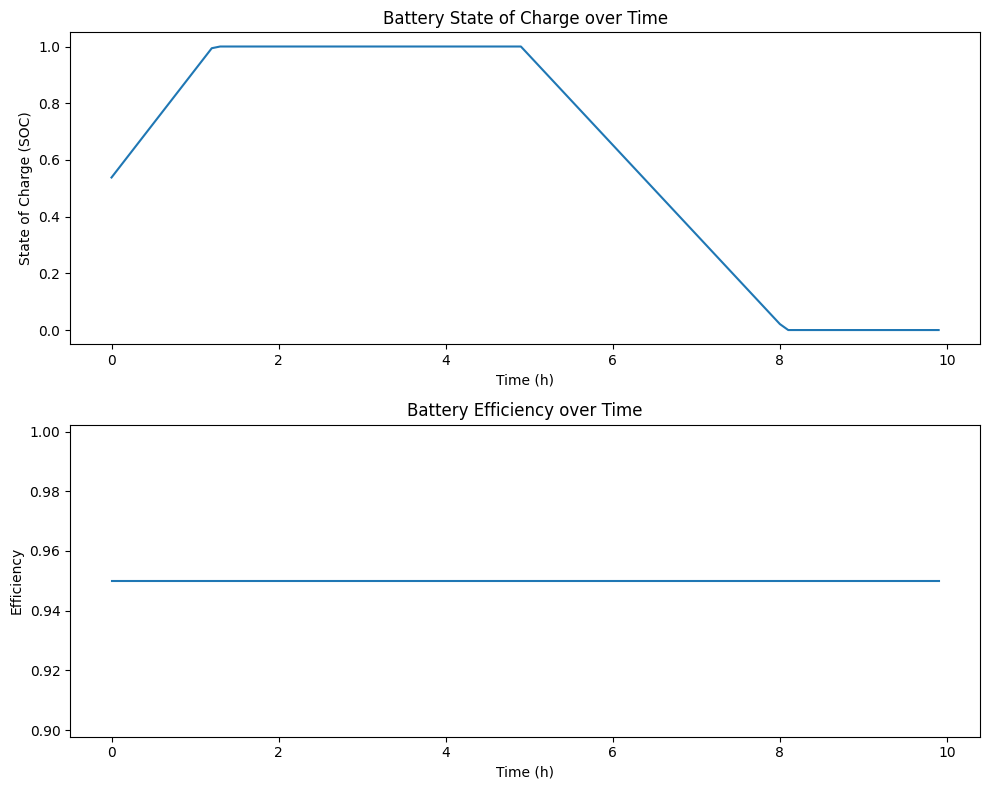
axes[1].set\_title('Battery Efficiency over Time')

# Adjust layout and display plots

plt.tight\_layout()

plt.show()

**OUTPUT:**



RESULT:

**Simulation Results of Li-ion Battery**

* 1. Initial Condition: Battery Capacity = **50 Ah**
  2. Initial SOC = **50% (0.5)**
  3. Voltage = **350 V (assumed constant)**
  4. **Charging Phase (0 – 5 hours, Current = +20 A)**

SOC gradually increased due to charging with 95% efficiency.

* 1. SOC reached **100% (fully charged)** in approximately **4.5 hours**, and then remained constant at 1.0 till 5 hours.
  2. Charging efficiency throughout = 0.95.
  3. Battery discharged with efficiency of 95%.
  4. SOC decreased steadily after 5 hours.
  5. At the end of 10 hours, SOC dropped to approximately **0.48 (48%)**.
  6. Discharging efficiency throughout = 0.95.
  7. Voltage assumed constant at **350 V** (no dynamic variation in this simplified model).
  8. The battery was successfully charged from 50% SOC to full in the first half of the simulation.
  9. In the second half, discharging reduced the SOC back close to its starting point (~48%), confirming that the simulation is consistent with given charging/discharging currents and efficiencies.

1. **Discharging Phase (5 – 10 hours, Current = –15 A)**
2. Voltage Behavior
3. Overall Observation

Conclusion:

 The project explains how a battery stores energy during charging and supplies it during discharging.

 It helps to understand important factors like battery efficiency, capacity, and lifespan.

 Proper charging and discharging methods improve battery safety and performance.

 This knowledge is very useful in mobile devices, electric vehicles, and renewable energy systems.

 Overall, the project highlights the importance of batteries in modern technology.