

Supplementary Files for RL2: Reinforce Large Language Model to Assist Safe Reinforcement Learning for Energy Management of Active Distribution Networks

I. COMPLETE SYSTEM PROMPTS

You are an expert in power system operation and reinforcement learning. I want to design a voltage penalty function and a branch power penalty function for a safe deep reinforcement learning task in energy management of the power system.

Environment Description

I have a 33-bus distribution network system. Each bus in the system has its corresponding loads. And 2 diesels are installed in bus-18 and bus-33, 2 batteries are installed in bus-21 and bus-24, 2 photovoltaic (PV) inverters are installed in bus-22 and bus-25. The active power of diesels, batteries and reactive power of diesels, PV inverters can be adjusted to minimize operational cost.

Task Description

Facing the variations in loads and PV generation, this energy management task aims to adjust the active power of diesels, batteries and reactive power of diesels, PV inverters to minimize the operational cost of the distribution network system. In addition, the voltage magnitude of each bus must not exceed the upper and lower limits. The upper limit is 1.05 p.u., and the lower limit is 0.95 p.u.. The power of each branch must not exceed the capacity. The capacity is 5.0 MVA.

1. The safety requirements must be first satisfied, i.e., the voltage magnitude of each bus must not exceed the upper and lower limits, and the power of each branch must not exceed the capacity.
2. After the safety requirements are satisfied, the operational cost should be minimized.

Output Format

```
```python
```

```
def calculate_voltage_penalty(self, voltage):
```

```
 voltage_penalty = ## your designed voltage penalty function using the voltage
 return voltage_penalty
```

```
def calculate_branch_power_penalty(self, branch_power):
```

```
 branch_power_penalty = ## your designed branch power penalty function using the branch power
 return branch_power_penalty
```

```
```
```

Here, voltage is the voltage magnitude of a specific bus, branch_power is the branch power of a specific branch.

Reward Calculation

After the design of the voltage penalty function and the branch power penalty function, the reward will be calculated as:

```
voltage_penalty = np.mean(self.calculate_voltage_penalty(voltage))
```

```
branch_power_penalty = np.mean(self.calculate_branch_power_penalty(branch_power))
```

```
reward = -operational_cost * 2.0 - voltage_penalty - branch_power_penalty
```

Here, operational_cost is the operational cost of the distribution network system.

Voltage Penalty and Branch Power Penalty Function Requirements

1. The voltage penalty function must limit the voltage magnitude in the safe range (not higher than 1.05 p.u. and not lower than 0.95 p.u.).
2. The branch power penalty function must limit the branch power in the safe range (not higher than 5.0 MVA).
3. To balance the safety requirements and operational cost, the value should not be too large or too small.
4. The pattern of the penalty functions should be simple.

Rules

1. You must only use the voltage and the branch power to calculate the penalty.
2. You must follow the output format.
3. You must consider the task and requirements.

You are an expert in power system operation and reinforcement learning. I want to design a voltage penalty function and a branch power penalty function for a safe deep reinforcement learning task in energy management of the power system.

Environment Description

I have a 69-bus distribution network system. Each bus in the system has its corresponding loads. And 2 diesels are installed in bus-18 and bus-58, 2 batteries are installed in bus-34 and bus-45, 2 photovoltaic (PV) inverters are installed in bus-35 and bus-46. The active power of diesels, batteries and reactive power of diesels, PV inverters can be adjusted to minimize operational cost.

Task Description

Facing the variations in loads and PV generation, this energy management task aims to adjust the active power of diesels, batteries and reactive power of diesels, PV inverters to minimize the operational cost of the distribution network system. In addition, the voltage magnitude of each bus must not exceed the upper and lower limits. The upper limit is 1.05 p.u., and the lower limit is 0.95 p.u.. The power of each branch must not exceed the capacity. The capacity is 4.7 MVA.

1. The safety requirements must be first satisfied, i.e., the voltage magnitude of each bus must not exceed the upper and lower limits, and the power of each branch must not exceed the capacity.
2. After the safety requirements are satisfied, the operational cost should be minimized.

Output Format

```
```python
def calculate_voltage_penalty(self, voltage):
 voltage_penalty = ## your designed voltage penalty function using the voltage
 return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
 branch_power_penalty = ## your designed branch power penalty function using the branch power
 return branch_power_penalty
...```
```

Here, voltage is the voltage magnitude of a specific bus, branch\_power is the branch power of a specific branch.

#### # Reward Calculation

After the design of the voltage penalty function and the branch power penalty function, the reward will be calculated as:

```
voltage_penalty = np.mean(self.calculate_voltage_penalty(voltage))
branch_power_penalty = np.mean(self.calculate_branch_power_penalty(branch_power))
reward = -operational_cost - voltage_penalty - branch_power_penalty
```

Here, operational\_cost is the operational cost of the distribution network system.

#### # Voltage Penalty and Branch Power Penalty Function Requirements

1. The voltage penalty function must limit the voltage magnitude in the safe range (not higher than 1.05 p.u. and not lower than 0.95 p.u.).
2. The branch power penalty function must limit the branch power in the safe range (not higher than 4.7 MVA).
3. To balance the safety requirements and operational cost, the value should not be too large or too small.
4. The pattern of the penalty functions should be simple.

#### # Rules

1. You must only use the voltage and the branch power to calculate the penalty.
2. You must follow the output format.
3. You must consider the task and requirements.

## II. COMPLETE REINFORCEMENT PROMPTS

The designed penalty functions are:

```
```python[code]
```
```

The performance of the RL agent with the designed voltage penalty function and branch power penalty function is [violation times >20bad, 10~20not good, <10good].

# Training Results

The training lasts 500 episodes, the reward converges to [reward\_converge].

# Test Results

The average operational cost is [average\_operational\_cost]\$.

The average voltage violation rate is [average\_voltage\_violation\_rate]%.

The average branch power violation rate is [average\_branch\_power\_violation\_rate]%.

The voltage violates [voltage\_violation] times in one day.

The branch power violates [branch\_power\_violation] times in one day.

Redesign the voltage penalty function and branch power penalty function based on the given feedback. Let's think step by step:  
1. Check current voltage penalty function and branch power penalty function and test results to see if the RL agent performs well.

2. If the performance is still bad, you can design a new voltage penalty function pattern and branch power penalty function pattern to improve the performance.

3. If the voltage or branch power violates many times, you can increase the value of the voltage penalty function or branch power penalty function by multiplying current function with a larger scaling factor.

4. If the voltage or branch power violates few times, you can decrease the value of the voltage penalty function and branch power penalty function by multiplying current function with a smaller scaling factor, so that the operational cost can be minimized.

## III. DIALOGUE PROCESS OF 33-BUS CASE

### ***Begin: Dialogue 1, LLM Agent k = 0***

```
```python
def calculate_voltage_penalty(self, voltage):
    k_v = 100 # Penalty factor for voltage
    if voltage > 1.05:
        voltage_penalty = k_v * (voltage - 1.05)**2
    elif voltage < 0.95:
        voltage_penalty = k_v * (voltage - 0.95)**2
    else:
        voltage_penalty = 0
    return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
    k_p = 100 # Penalty factor for branch power
    if branch_power > 5.0:
        branch_power_penalty = k_p * (branch_power - 5.0)**2
    else:
        branch_power_penalty = 0
    return branch_power_penalty
```
```

### Dialogue 1, ADN Operator, k = 1

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

#### # Training Results

The training lasts 500 episodes, the reward converges to -31.99.

#### # Test Results

The average operational cost is 15.97\$.

The average voltage violation rate is 7.891%.

The average branch power violation rate is 0.0%.

The voltage violates 37 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Dialogue 1, LLM Agent k = 1

```
```python
def calculate_voltage_penalty(self, voltage):
    k_v = 200 # Increased penalty factor for voltage
    if voltage > 1.05:
        voltage_penalty = k_v * (voltage - 1.05)**2
    elif voltage < 0.95:
        voltage_penalty = k_v * (voltage - 0.95)**2
    else:
        voltage_penalty = 0
    return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
    k_p = 50 # Reduced penalty factor for branch power
    if branch_power > 5.0:
        branch_power_penalty = k_p * (branch_power - 5.0)**2
    else:
        branch_power_penalty = 0
    return branch_power_penalty
```
```

### Dialogue 1, ADN Operator, k = 2

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

#### # Training Results

The training lasts 500 episodes, the reward converges to -29.65.

#### # Test Results

The average operational cost is 14.71\$.

The average voltage violation rate is 9.249%.

The average branch power violation rate is 0.03255%.

The voltage violates 41 times in one day.

The branch power violates 1 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Dialogue 1, LLM Agent k = 2

```

```python
def calculate_voltage_penalty(self, voltage):
    k_v = 500 # Increased penalty factor for voltage
    if voltage > 1.05:
        voltage_penalty = k_v * (voltage - 1.05)
    elif voltage < 0.95:
        voltage_penalty = k_v * (0.95 - voltage)
    else:
        voltage_penalty = 0
    return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
    k_p = 200 # Increased penalty factor for branch power
    if branch_power > 5.0:
        branch_power_penalty = k_p * (branch_power - 5.0)
    else:
        branch_power_penalty = 0
    return branch_power_penalty
```

```

### Dialogue 1, ADN Operator, k = 3

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is good.

#### # Training Results

The training lasts 500 episodes, the reward converges to -33.76.

#### # Test Results

The average operational cost is 16.75\$.

The average voltage violation rate is 0.3157%.

The average branch power violation rate is 0.0%.

The voltage violates 9 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Dialogue 1, LLM Agent k = 3

```

'''python
def calculate_voltage_penalty(self, voltage):
 k_v = 400 # Slightly reduced penalty factor for voltage
 if voltage > 1.05:
 voltage_penalty = k_v * (voltage - 1.05)
 elif voltage < 0.95:
 voltage_penalty = k_v * (0.95 - voltage)
 else:
 voltage_penalty = 0
 return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
 k_p = 150 # Slightly reduced penalty factor for branch power
 if branch_power > 5.0:
 branch_power_penalty = k_p * (branch_power - 5.0)
 else:
 branch_power_penalty = 0
 return branch_power_penalty
'''

```

### Dialogue 1, ADN Operator, k = 4

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is not good.

#### # Training Results

The training lasts 500 episodes, the reward converges to -30.73.

#### # Test Results

The average operational cost is 14.99\$.

The average voltage violation rate is 0.4104%.

The average branch power violation rate is 0.0%.

The voltage violates 11 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Dialogue 1, LLM Agent k = 4

```

```python
def calculate_voltage_penalty(self, voltage):
    k_v = 450 # Slightly increased penalty factor for voltage
    if voltage > 1.05:
        voltage_penalty = k_v * (voltage - 1.05)
    elif voltage < 0.95:
        voltage_penalty = k_v * (0.95 - voltage)
    else:
        voltage_penalty = 0
    return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
    k_p = 120 # Slightly reduced penalty factor for branch power
    if branch_power > 5.0:
        branch_power_penalty = k_p * (branch_power - 5.0)
    else:
        branch_power_penalty = 0
    return branch_power_penalty
```

```

### Dialogue 1, ADN Operator, k = 5

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is good.

#### # Training Results

The training lasts 500 episodes, the reward converges to -30.05.

#### # Test Results

The average operational cost is 14.84\$.

The average voltage violation rate is 0.3472%.

The average branch power violation rate is 0.0%.

The voltage violates 8 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Dialogue 1, LLM Agent k = 5

```

python
def calculate_voltage_penalty(self, voltage):
 k_v = 400 # Slightly reduced penalty factor for voltage
 if voltage > 1.05:
 voltage_penalty = k_v * (voltage - 1.05)
 elif voltage < 0.95:
 voltage_penalty = k_v * (0.95 - voltage)
 else:
 voltage_penalty = 0
 return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
 k_p = 100 # Further reduced penalty factor for branch power
 if branch_power > 5.0:
 branch_power_penalty = k_p * (branch_power - 5.0)
 else:
 branch_power_penalty = 0
 return branch_power_penalty

```

### End: Dialogue 1, ADN Operator, k = 6

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
The performance of the RL agent with the designed voltage penalty function and branch power penalty function is good.

#### # Training Results

The training lasts 500 episodes, the reward converges to -30.49.

#### # Test Results

The average operational cost is 15.04\$.

The average voltage violation rate is 0.2841%.

The average branch power violation rate is 0.0%.

The voltage violates 7 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)



### Begin: Dialogue 2, LLM Agent k = 0

```

```python
def calculate_voltage_penalty(self, voltage):
    k_v = 100 # Scaling factor for voltage penalty
    if voltage > 1.05:
        voltage_penalty = k_v * (voltage - 1.05)**2
    elif voltage < 0.95:
        voltage_penalty = k_v * (voltage - 0.95)**2
    else:
        voltage_penalty = 0
    return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
    k_p = 100 # Scaling factor for branch power penalty
    if branch_power > 5.0:
        branch_power_penalty = k_p * (branch_power - 5.0)**2
    else:
        branch_power_penalty = 0
    return branch_power_penalty
```

```

### Dialogue 2, ADN Operator, k = 1

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

#### # Training Results

The training lasts 500 episodes, the reward converges to -30.10.

#### # Test Results

The average operational cost is 14.97\$.

The average voltage violation rate is 11.58%.

The average branch power violation rate is 0.0%.

The voltage violates 39 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

## Dialogue 2, LLM Agent k = 1

```

```python
def calculate_voltage_penalty(self, voltage):
    k_v = 500 # Increased scaling factor for voltage penalty
    if voltage > 1.05:
        voltage_penalty = k_v * (voltage - 1.05)**2 + 100 * (voltage - 1.05)
    elif voltage < 0.95:
        voltage_penalty = k_v * (voltage - 0.95)**2 + 100 * (0.95 - voltage)
    else:
        voltage_penalty = 0
    return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
    k_p = 100 # Scaling factor for branch power penalty
    if branch_power > 5.0:
        branch_power_penalty = k_p * (branch_power - 5.0)**2
    else:
        branch_power_penalty = 0
    return branch_power_penalty
```

```

## Dialogue 2, ADN Operator, k = 2

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is not good.

### # Training Results

The training lasts 500 episodes, the reward converges to -31.68.

### # Test Results

The average operational cost is 15.25\$.

The average voltage violation rate is 0.9785%.

The average branch power violation rate is 0.03255%.

The voltage violates 17 times in one day.

The branch power violates 1 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

## Dialogue 2, LLM Agent k = 2

```

python
def calculate_voltage_penalty(self, voltage):
 k_v = 1000 # Increased scaling factor for voltage penalty
 if voltage > 1.05:
 voltage_penalty = k_v * (voltage - 1.05)**2 + 200 * (voltage - 1.05)
 elif voltage < 0.95:
 voltage_penalty = k_v * (voltage - 0.95)**2 + 200 * (0.95 - voltage)
 else:
 voltage_penalty = 0
 return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
 k_p = 200 # Increased scaling factor for branch power penalty
 if branch_power > 5.0:
 branch_power_penalty = k_p * (branch_power - 5.0)**2 + 100 * (branch_power - 5.0)
 else:
 branch_power_penalty = 0
 return branch_power_penalty

```

## Dialogue 2, ADN Operator, k = 3

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is not good.

### # Training Results

The training lasts 500 episodes, the reward converges to -34.71.

### # Test Results

The average operational cost is 16.09\$.

The average voltage violation rate is 0.8207%.

The average branch power violation rate is 0.03255%.

The voltage violates 15 times in one day.

The branch power violates 1 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Dialogue 2, LLM Agent k = 3

```

```python
def calculate_voltage_penalty(self, voltage):
    k_v = 500 # Reduced scaling factor for voltage penalty
    if voltage > 1.05:
        voltage_penalty = k_v * (voltage - 1.05)**2 + 100 * (voltage - 1.05)
    elif voltage < 0.95:
        voltage_penalty = k_v * (voltage - 0.95)**2 + 100 * (0.95 - voltage)
    else:
        voltage_penalty = 0
    return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
    k_p = 100 # Reduced scaling factor for branch power penalty
    if branch_power > 5.0:
        branch_power_penalty = k_p * (branch_power - 5.0)**2 + 50 * (branch_power - 5.0)
    else:
        branch_power_penalty = 0
    return branch_power_penalty
```

```

### Dialogue 2, ADN Operator, k = 4

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is not good.

#### # Training Results

The training lasts 500 episodes, the reward converges to -31.01.

#### # Test Results

The average operational cost is 15.18\$.

The average voltage violation rate is 0.8523%.

The average branch power violation rate is 0.0%.

The voltage violates 16 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Dialogue 2, LLM Agent k = 4

```

python
def calculate_voltage_penalty(self, voltage):
 k_v = 300 # Adjusted scaling factor for voltage penalty
 if voltage > 1.05:
 voltage_penalty = k_v * (voltage - 1.05)**2 + 150 * (voltage - 1.05) + 50
 elif voltage < 0.95:
 voltage_penalty = k_v * (voltage - 0.95)**2 + 150 * (0.95 - voltage) + 50
 else:
 voltage_penalty = 0
 return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
 k_p = 150 # Adjusted scaling factor for branch power penalty
 if branch_power > 5.0:
 branch_power_penalty = k_p * (branch_power - 5.0)**2 + 75 * (branch_power - 5.0) + 25
 else:
 branch_power_penalty = 0
 return branch_power_penalty

```

### Dialogue 2, ADN Operator, k = 5

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is good.

#### # Training Results

The training lasts 500 episodes, the reward converges to -32.81.

#### # Test Results

The average operational cost is 16.40\$.

The average voltage violation rate is 0.0%.

The average branch power violation rate is 0.0%.

The voltage violates 0 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Dialogue 2, LLM Agent k = 5

```

python
def calculate_voltage_penalty(self, voltage):
 k_v = 200 # Further reduced scaling factor for voltage penalty
 if voltage > 1.05:
 voltage_penalty = k_v * (voltage - 1.05)**2 + 100 * (voltage - 1.05) + 25
 elif voltage < 0.95:
 voltage_penalty = k_v * (voltage - 0.95)**2 + 100 * (0.95 - voltage) + 25
 else:
 voltage_penalty = 0
 return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
 k_p = 100 # Further reduced scaling factor for branch power penalty
 if branch_power > 5.0:
 branch_power_penalty = k_p * (branch_power - 5.0)**2 + 50 * (branch_power - 5.0) + 10
 else:
 branch_power_penalty = 0
 return branch_power_penalty

```

### End: Dialogue 2, ADN Operator, k = 6

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is good.

#### # Training Results

The training lasts 500 episodes, the reward converges to -31.28.

#### # Test Results

The average operational cost is 15.64\$.

The average voltage violation rate is 0.0%.

The average branch power violation rate is 0.0%.

The voltage violates 0 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Begin: Dialogue 3, LLM Agent k = 0

```

``python
def calculate_voltage_penalty(self, voltage):
 # Define the upper and lower limits
 v_upper_limit = 1.05
 v_lower_limit = 0.95

 # Calculate the deviation from the safe range
 if voltage > v_upper_limit:
 deviation = voltage - v_upper_limit
 elif voltage < v_lower_limit:
 deviation = v_lower_limit - voltage
 else:
 deviation = 0.0

 # Quadratic penalty for the deviation
 voltage_penalty = deviation ** 2

 return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
 # Define the power limit
 power_limit = 5.0 # in MVA

 # Calculate the deviation from the safe limit
 if branch_power > power_limit:
 deviation = branch_power - power_limit
 else:
 deviation = 0.0

 # Quadratic penalty for the deviation
 branch_power_penalty = deviation ** 2

 return branch_power_penalty

```

### Dialogue 3, ADN Operator, k = 1

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

#### # Training Results

The training lasts 500 episodes, the reward converges to -29.57.

#### # Test Results

The average operational cost is 14.78\$.

The average voltage violation rate is 11.90%.

The average branch power violation rate is 0.03255%.

The voltage violates 41 times in one day.

The branch power violates 1 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Dialogue 3, LLM Agent k = 1

```

``python
def calculate_voltage_penalty(self, voltage):
 # Define the upper and lower limits
 v_upper_limit = 1.05
 v_lower_limit = 0.95

 # Calculate the deviation from the safe range
 if voltage > v_upper_limit:
 deviation = voltage - v_upper_limit
 elif voltage < v_lower_limit:
 deviation = v_lower_limit - voltage
 else:
 deviation = 0.0

 # Cubic penalty for the deviation with a scaling factor
 voltage_penalty = 10 * (deviation ** 3)

 return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
 # Define the power limit
 power_limit = 5.0 # in MVA

 # Calculate the deviation from the safe limit
 if branch_power > power_limit:
 deviation = branch_power - power_limit
 else:
 deviation = 0.0

 # Quadratic penalty for the deviation with a scaling factor
 branch_power_penalty = 0.1 * (deviation ** 2)

 return branch_power_penalty

```

### Dialogue 3, ADN Operator, k = 2

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

#### # Training Results

The training lasts 500 episodes, the reward converges to -29.45.

#### # Test Results

The average operational cost is 14.73\$.

The average voltage violation rate is 11.96%.

The average branch power violation rate is 0.1953%.

The voltage violates 41 times in one day.

The branch power violates 6 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)



### Dialogue 3, LLM Agent k = 2

```

```python
def calculate_voltage_penalty(self, voltage):
    # Define the upper and lower limits
    v_upper_limit = 1.05
    v_lower_limit = 0.95

    # Calculate the deviation from the safe range
    if voltage > v_upper_limit:
        deviation = voltage - v_upper_limit
    elif voltage < v_lower_limit:
        deviation = v_lower_limit - voltage
    else:
        deviation = 0.0

    # Cubic penalty for the deviation with a larger scaling factor
    voltage_penalty = 100 * (deviation ** 3)

    return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
    # Define the power limit
    power_limit = 5.0 # in MVA

    # Calculate the deviation from the safe limit
    if branch_power > power_limit:
        deviation = branch_power - power_limit
    else:
        deviation = 0.0

    # Quadratic penalty for the deviation with a larger scaling factor
    branch_power_penalty = 10 * (deviation ** 2)

    return branch_power_penalty

```

Dialogue 3, ADN Operator, k = 3

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

Training Results

The training lasts 500 episodes, the reward converges to -29.43.

Test Results

The average operational cost is 14.71\$.

The average voltage violation rate is 11.96%.

The average branch power violation rate is 0.03255%.

The voltage violates 42 times in one day.

The branch power violates 1 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

Dialogue 3, LLM Agent k = 3

```

```python
def calculate_voltage_penalty(self, voltage):
 # Define the upper and lower limits
 v_upper_limit = 1.05
 v_lower_limit = 0.95

 # Calculate the deviation from the safe range
 if voltage > v_upper_limit:
 deviation = voltage - v_upper_limit
 elif voltage < v_lower_limit:
 deviation = v_lower_limit - voltage
 else:
 deviation = 0.0

 # Cubic penalty for the deviation with a larger scaling factor
 voltage_penalty = 500 * (deviation ** 3)

 return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
 # Define the power limit
 power_limit = 5.0 # in MVA

 # Calculate the deviation from the safe limit
 if branch_power > power_limit:
 deviation = branch_power - power_limit
 else:
 deviation = 0.0

 # Quadratic penalty for the deviation with a slightly reduced scaling factor
 branch_power_penalty = 5 * (deviation ** 2)

 return branch_power_penalty

```

### Dialogue 3, ADN Operator, k = 4

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

#### # Training Results

The training lasts 500 episodes, the reward converges to -29.30.

#### # Test Results

The average operational cost is 14.64\$.

The average voltage violation rate is 11.52%.

The average branch power violation rate is 0.03255%.

The voltage violates 40 times in one day.

The branch power violates 1 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Dialogue 3, LLM Agent k = 4

```

```python
def calculate_voltage_penalty(self, voltage):
    # Define the upper and lower limits
    v_upper_limit = 1.05
    v_lower_limit = 0.95

    # Calculate the deviation from the safe range
    if voltage > v_upper_limit:
        deviation = voltage - v_upper_limit
    elif voltage < v_lower_limit:
        deviation = v_lower_limit - voltage
    else:
        deviation = 0.0

    # Cubic penalty for the deviation with a larger scaling factor
    voltage_penalty = 1000 * (deviation ** 3)

    return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
    # Define the power limit
    power_limit = 5.0 # in MVA

    # Calculate the deviation from the safe limit
    if branch_power > power_limit:
        deviation = branch_power - power_limit
    else:
        deviation = 0.0

    # Quadratic penalty for the deviation with a smaller scaling factor
    branch_power_penalty = 2 * (deviation ** 2)

    return branch_power_penalty

```

Dialogue 3, ADN Operator, k = 5

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

Training Results

The training lasts 500 episodes, the reward converges to -29.53.

Test Results

The average operational cost is 14.75\$.

The average voltage violation rate is 11.08%.

The average branch power violation rate is 0.03255%.

The voltage violates 41 times in one day.

The branch power violates 1 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

Dialogue 3, LLM Agent k = 5

```

```python
def calculate_voltage_penalty(self, voltage):
 # Define the upper and lower limits
 v_upper_limit = 1.05
 v_lower_limit = 0.95

 # Calculate the deviation from the safe range
 if voltage > v_upper_limit:
 deviation = voltage - v_upper_limit
 elif voltage < v_lower_limit:
 deviation = v_lower_limit - voltage
 else:
 deviation = 0.0

 # Combined linear and quadratic penalty for the deviation with scaling factors
 voltage_penalty = 100 * (deviation ** 2) + 500 * deviation

 return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
 # Define the power limit
 power_limit = 5.0 # in MVA

 # Calculate the deviation from the safe limit
 if branch_power > power_limit:
 deviation = branch_power - power_limit
 else:
 deviation = 0.0

 # Combined linear and quadratic penalty for the deviation with scaling factors
 branch_power_penalty = 10 * (deviation ** 2) + 50 * deviation

 return branch_power_penalty

```

### Dialogue 3, ADN Operator, k = 6

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
The performance of the RL agent with the designed voltage penalty function and branch power penalty function is good.

#### # Training Results

The training lasts 500 episodes, the reward converges to -29.31.

#### # Test Results

The average operational cost is 14.65\$.

The average voltage violation rate is 0.03157%.

The average branch power violation rate is 0.0%.

The voltage violates 1 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Dialogue 3, LLM Agent k = 6

```

python
def calculate_voltage_penalty(self, voltage):
 # Define the upper and lower limits
 v_upper_limit = 1.05
 v_lower_limit = 0.95

 # Calculate the deviation from the safe range
 if voltage > v_upper_limit:
 deviation = voltage - v_upper_limit
 elif voltage < v_lower_limit:
 deviation = v_lower_limit - voltage
 else:
 deviation = 0.0

 # Combined linear and quadratic penalty for the deviation with slightly reduced scaling factors
 voltage_penalty = 80 * (deviation ** 2) + 400 * deviation

 return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
 # Define the power limit
 power_limit = 5.0 # in MVA

 # Calculate the deviation from the safe limit
 if branch_power > power_limit:
 deviation = branch_power - power_limit
 else:
 deviation = 0.0

 # Combined linear and quadratic penalty for the deviation with slightly reduced scaling factors
 branch_power_penalty = 8 * (deviation ** 2) + 40 * deviation

 return branch_power_penalty

```

### End: Dialogue 3, ADN Operator, k = 7

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is good.

#### # Training Results

The training lasts 500 episodes, the reward converges to -29.83.

#### # Test Results

The average operational cost is 14.85\$.

The average voltage violation rate is 0.2525%.

The average branch power violation rate is 0.0%.

The voltage violates 8 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Begin: Dialogue 4, LLM Agent k = 0

```

```python
def calculate_voltage_penalty(self, voltage):
    # Define the upper and lower limits for the voltage
    upper_limit = 1.05
    lower_limit = 0.95

    # Calculate the deviation from the safe range
    if voltage > upper_limit:
        deviation = voltage - upper_limit
    elif voltage < lower_limit:
        deviation = lower_limit - voltage
    else:
        deviation = 0.0

    # Use a quadratic penalty function
    voltage_penalty = deviation ** 2

    return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
    # Define the capacity limit for the branch power
    capacity_limit = 5.0 # in MVA

    # Calculate the deviation from the capacity limit
    if branch_power > capacity_limit:
        deviation = branch_power - capacity_limit
    else:
        deviation = 0.0

    # Use a quadratic penalty function
    branch_power_penalty = deviation ** 2

    return branch_power_penalty
```

```

### Dialogue 4, ADN Operator, k = 1

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

#### # Training Results

The training lasts 300 episodes, the reward converges to -29.41.

#### # Test Results

The average operational cost is 14.70\$.

The average voltage violation rate is 12.25%.

The average branch power violation rate is 0.03255%.

The voltage violates 40 times in one day.

The branch power violates 1 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Dialogue 4, LLM Agent k = 1

```

```python
def calculate_voltage_penalty(self, voltage):
    # Define the upper and lower limits for the voltage
    upper_limit = 1.05
    lower_limit = 0.95

    # Calculate the deviation from the safe range
    if voltage > upper_limit:
        deviation = voltage - upper_limit
    elif voltage < lower_limit:
        deviation = lower_limit - voltage
    else:
        deviation = 0.0

    # Use a quadratic penalty function with a higher scaling factor
    voltage_penalty = (deviation ** 2) * 1000 # Increased scaling factor to 1000

    return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
    # Define the capacity limit for the branch power
    capacity_limit = 5.0 # in MVA

    # Calculate the deviation from the capacity limit
    if branch_power > capacity_limit:
        deviation = branch_power - capacity_limit
    else:
        deviation = 0.0

    # Use a quadratic penalty function with a moderate scaling factor
    branch_power_penalty = (deviation ** 2) * 100 # Moderate scaling factor to 100

    return branch_power_penalty
```

```

### Dialogue 4, ADN Operator, k = 2

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

#### # Training Results

The training lasts 300 episodes, the reward converges to -30.69.

#### # Test Results

The average operational cost is 15.19\$.

The average voltage violation rate is 3.914%.

The average branch power violation rate is 0.0%.

The voltage violates 31 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Dialogue 4, LLM Agent k = 2

```

```python
def calculate_voltage_penalty(self, voltage):
    # Define the upper and lower limits for the voltage
    upper_limit = 1.05
    lower_limit = 0.95

    # Calculate the deviation from the safe range
    if voltage > upper_limit:
        deviation = voltage - upper_limit
    elif voltage < lower_limit:
        deviation = lower_limit - voltage
    else:
        deviation = 0.0

    # Use a piecewise quadratic penalty function with a higher scaling factor
    if deviation > 0.05:
        voltage_penalty = (deviation ** 2) * 5000 # Higher scaling factor for large deviations
    else:
        voltage_penalty = (deviation ** 2) * 1000 # Moderate scaling factor for small deviations

    return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
    # Define the capacity limit for the branch power
    capacity_limit = 5.0 # in MVA

    # Calculate the deviation from the capacity limit
    if branch_power > capacity_limit:
        deviation = branch_power - capacity_limit
    else:
        deviation = 0.0

    # Use a quadratic penalty function with a slightly reduced scaling factor
    branch_power_penalty = (deviation ** 2) * 50 # Slightly reduced scaling factor to 50

    return branch_power_penalty
```

```

### Dialogue 4, ADN Operator, k = 3

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

#### # Training Results

The training lasts 300 episodes, the reward converges to -30.80.

#### # Test Results

The average operational cost is 15.22\$.

The average voltage violation rate is 4.167%.

The average branch power violation rate is 0.03255%.

The voltage violates 35 times in one day.

The branch power violates 1 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)



### Dialogue 4, LLM Agent k = 3

```

```python
def calculate_voltage_penalty(self, voltage):
    # Define the upper and lower limits for the voltage
    upper_limit = 1.05
    lower_limit = 0.95

    # Calculate the deviation from the safe range
    if voltage > upper_limit:
        deviation = voltage - upper_limit
    elif voltage < lower_limit:
        deviation = lower_limit - voltage
    else:
        deviation = 0.0

    # Use a piecewise quadratic penalty function with increased scaling factors
    if deviation > 0.05:
        voltage_penalty = (deviation ** 2) * 10000 # Very high scaling factor for large deviations
    else:
        voltage_penalty = (deviation ** 2) * 2000 # Higher scaling factor for small deviations

    return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
    # Define the capacity limit for the branch power
    capacity_limit = 5.0 # in MVA

    # Calculate the deviation from the capacity limit
    if branch_power > capacity_limit:
        deviation = branch_power - capacity_limit
    else:
        deviation = 0.0

    # Use a quadratic penalty function with a higher scaling factor
    branch_power_penalty = (deviation ** 2) * 100 # Increased scaling factor to 100

    return branch_power_penalty
```

```

### Dialogue 4, ADN Operator, k = 4

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

#### # Training Results

The training lasts 300 episodes, the reward converges to -30.32.

#### # Test Results

The average operational cost is 15.00\$.

The average voltage violation rate is 2.746%.

The average branch power violation rate is 0.0%.

The voltage violates 28 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Dialogue 4, LLM Agent k = 4

```

```python
def calculate_voltage_penalty(self, voltage):
    # Define the upper and lower limits for the voltage
    upper_limit = 1.05
    lower_limit = 0.95

    # Calculate the deviation from the safe range
    if voltage > upper_limit:
        deviation = voltage - upper_limit
    elif voltage < lower_limit:
        deviation = lower_limit - voltage
    else:
        deviation = 0.0

    # Use a piecewise quadratic penalty function with increased scaling factors
    if deviation > 0.05:
        voltage_penalty = (deviation ** 2) * 20000 # Very high scaling factor for large deviations
    else:
        voltage_penalty = (deviation ** 2) * 3000 # Higher scaling factor for small deviations

    return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
    # Define the capacity limit for the branch power
    capacity_limit = 5.0 # in MVA

    # Calculate the deviation from the capacity limit
    if branch_power > capacity_limit:
        deviation = branch_power - capacity_limit
    else:
        deviation = 0.0

    # Use a quadratic penalty function with a slightly reduced scaling factor
    branch_power_penalty = (deviation ** 2) * 75 # Slightly reduced scaling factor to 75

    return branch_power_penalty
```

```

### Dialogue 4, ADN Operator, k = 5

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

#### # Training Results

The training lasts 300 episodes, the reward converges to -31.19.

#### # Test Results

The average operational cost is 15.25\$.

The average voltage violation rate is 2.052%.

The average branch power violation rate is 0.03255%.

The voltage violates 27 times in one day.

The branch power violates 1 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Dialogue 4, LLM Agent k = 5

```

'''python
def calculate_voltage_penalty(self, voltage):
 # Define the upper and lower limits for the voltage
 upper_limit = 1.05
 lower_limit = 0.95

 # Calculate the deviation from the safe range
 if voltage > upper_limit:
 deviation = voltage - upper_limit
 elif voltage < lower_limit:
 deviation = lower_limit - voltage
 else:
 deviation = 0.0

 # Use a piecewise quadratic penalty function with increased scaling factors
 if deviation > 0.05:
 voltage_penalty = (deviation ** 2) * 25000 # Very high scaling factor for large deviations
 else:
 voltage_penalty = (deviation ** 2) * 4000 # Higher scaling factor for small deviations

 return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
 # Define the capacity limit for the branch power
 capacity_limit = 5.0 # in MVA

 # Calculate the deviation from the capacity limit
 if branch_power > capacity_limit:
 deviation = branch_power - capacity_limit
 else:
 deviation = 0.0

 # Use a quadratic penalty function with a slightly reduced scaling factor
 branch_power_penalty = (deviation ** 2) * 50 # Slightly reduced scaling factor to 50

 return branch_power_penalty
'''

```

### Dialogue 4, ADN Operator, k = 6

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

#### # Training Results

The training lasts 300 episodes, the reward converges to -30.31.

#### # Test Results

The average operational cost is 15.07\$.

The average voltage violation rate is 1.610%.

The average branch power violation rate is 0.0%.

The voltage violates 23 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Dialogue 4, LLM Agent k = 6

```

```python
def calculate_voltage_penalty(self, voltage):
    # Define the upper and lower limits for the voltage
    upper_limit = 1.05
    lower_limit = 0.95

    # Calculate the deviation from the safe range
    if voltage > upper_limit:
        deviation = voltage - upper_limit
    elif voltage < lower_limit:
        deviation = lower_limit - voltage
    else:
        deviation = 0.0

    # Use a piecewise linear penalty function with different scaling factors
    if deviation > 0.05:
        voltage_penalty = (deviation - 0.05) * 50000 + 2500 # Very high scaling factor for large deviations
    elif deviation > 0.01:
        voltage_penalty = (deviation - 0.01) * 10000 + 100 # Higher scaling factor for medium deviations
    else:
        voltage_penalty = (deviation ** 2) * 4000 # Quadratic penalty for small deviations

    return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
    # Define the capacity limit for the branch power
    capacity_limit = 5.0 # in MVA

    # Calculate the deviation from the capacity limit
    if branch_power > capacity_limit:
        deviation = branch_power - capacity_limit
    else:
        deviation = 0.0

    # Use a quadratic penalty function with a slightly reduced scaling factor
    branch_power_penalty = (deviation ** 2) * 50 # Slightly reduced scaling factor to 50

    return branch_power_penalty
```

```

### Dialogue 4, ADN Operator, k = 7

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
The performance of the RL agent with the designed voltage penalty function and branch power penalty function is good.

#### # Training Results

The training lasts 300 episodes, the reward converges to -42.61.

#### # Test Results

The average operational cost is 21.29\$.

The average voltage violation rate is 0.03157%.

The average branch power violation rate is 0.03255%.

The voltage violates 1 times in one day.

The branch power violates 1 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Dialogue 4, LLM Agent k = 7

```

```python
def calculate_voltage_penalty(self, voltage):
    # Define the upper and lower limits for the voltage
    upper_limit = 1.05
    lower_limit = 0.95

    # Calculate the deviation from the safe range
    if voltage > upper_limit:
        deviation = voltage - upper_limit
    elif voltage < lower_limit:
        deviation = lower_limit - voltage
    else:
        deviation = 0.0

    # Use a piecewise linear penalty function with different scaling factors
    if deviation > 0.05:
        voltage_penalty = (deviation - 0.05) * 20000 + 1000 # Reduced scaling factor for large deviations
    elif deviation > 0.01:
        voltage_penalty = (deviation - 0.01) * 5000 + 50 # Reduced scaling factor for medium deviations
    else:
        voltage_penalty = (deviation ** 2) * 2000 # Smaller quadratic penalty for small deviations

    return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
    # Define the capacity limit for the branch power
    capacity_limit = 5.0 # in MVA

    # Calculate the deviation from the capacity limit
    if branch_power > capacity_limit:
        deviation = branch_power - capacity_limit
    else:
        deviation = 0.0

    # Use a quadratic penalty function with a reduced scaling factor
    branch_power_penalty = (deviation ** 2) * 30 # Further reduced scaling factor to 30

    return branch_power_penalty
```

```

### Dialogue 4, ADN Operator, k = 8

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is good.

#### # Training Results

The training lasts 300 episodes, the reward converges to -33.43.

#### # Test Results

The average operational cost is 16.71\$.

The average voltage violation rate is 0.06313%.

The average branch power violation rate is 0.0%.

The voltage violates 1 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Dialogue 4, LLM Agent k = 8

```

```python
def calculate_voltage_penalty(self, voltage):
    # Define the upper and lower limits for the voltage
    upper_limit = 1.05
    lower_limit = 0.95

    # Calculate the deviation from the safe range
    if voltage > upper_limit:
        deviation = voltage - upper_limit
    elif voltage < lower_limit:
        deviation = lower_limit - voltage
    else:
        deviation = 0.0

    # Use a piecewise linear penalty function with different scaling factors
    if deviation > 0.05:
        voltage_penalty = (deviation - 0.05) * 15000 + 750 # Further reduced scaling factor for large deviations
    elif deviation > 0.01:
        voltage_penalty = (deviation - 0.01) * 4000 + 30 # Further reduced scaling factor for medium deviations
    else:
        voltage_penalty = (deviation ** 2) * 1500 # Smaller quadratic penalty for small deviations

    return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
    # Define the capacity limit for the branch power
    capacity_limit = 5.0 # in MVA

    # Calculate the deviation from the capacity limit
    if branch_power > capacity_limit:
        deviation = branch_power - capacity_limit
    else:
        deviation = 0.0

    # Use a quadratic penalty function with a reduced scaling factor
    branch_power_penalty = (deviation ** 2) * 20 # Further reduced scaling factor to 20

    return branch_power_penalty
```

```

### Dialogue 4, ADN Operator, k = 9

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

#### # Training Results

The training lasts 300 episodes, the reward converges to -31.95.

#### # Test Results

The average operational cost is 15.76\$.

The average voltage violation rate is 0.3157%.

The average branch power violation rate is 0.2279%.

The voltage violates 7 times in one day.

The branch power violates 7 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Dialogue 4, LLM Agent k = 9

```

```python
def calculate_voltage_penalty(self, voltage):
    # Define the upper and lower limits for the voltage
    upper_limit = 1.05
    lower_limit = 0.95

    # Calculate the deviation from the safe range
    if voltage > upper_limit:
        deviation = voltage - upper_limit
    elif voltage < lower_limit:
        deviation = lower_limit - voltage
    else:
        deviation = 0.0

    # Use a piecewise linear penalty function with different scaling factors
    if deviation > 0.05:
        voltage_penalty = (deviation - 0.05) * 30000 + 1500 # Increased scaling factor for large deviations
    elif deviation > 0.01:
        voltage_penalty = (deviation - 0.01) * 8000 + 60 # Increased scaling factor for medium deviations
    else:
        voltage_penalty = (deviation ** 2) * 3000 # Increased quadratic penalty for small deviations

    return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
    # Define the capacity limit for the branch power
    capacity_limit = 5.0 # in MVA

    # Calculate the deviation from the capacity limit
    if branch_power > capacity_limit:
        deviation = branch_power - capacity_limit
    else:
        deviation = 0.0

    # Use a quadratic penalty function with an increased scaling factor
    branch_power_penalty = (deviation ** 2) * 40 # Increased scaling factor to 40

    return branch_power_penalty
```

```

### End: Dialogue 4, ADN Operator, k = 10

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
The performance of the RL agent with the designed voltage penalty function and branch power penalty function is good.

#### # Training Results

The training lasts 300 episodes, the reward converges to -31.97.

#### # Test Results

The average operational cost is 15.99\$.

The average voltage violation rate is 0.0%.

The average branch power violation rate is 0.0%.

The voltage violates 0 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Begin: Dialogue 5, LLM Agent k = 0

```

```python
def calculate_voltage_penalty(self, voltage):
    k_v = 100 # Scaling factor
    if 0.95 <= voltage <= 1.05:
        voltage_penalty = 0
    elif voltage > 1.05:
        voltage_penalty = k_v * (voltage - 1.05)**2
    else: # voltage < 0.95
        voltage_penalty = k_v * (0.95 - voltage)**2
    return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
    k_p = 100 # Scaling factor
    if branch_power <= 5.0:
        branch_power_penalty = 0
    else: # branch_power > 5.0
        branch_power_penalty = k_p * (branch_power - 5.0)**2
    return branch_power_penalty
```

```

### Dialogue 5, ADN Operator, k = 1

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

#### # Training Results

The training lasts 500 episodes, the reward converges to -29.78.

#### # Test Results

The average operational cost is 14.85\$.

The average voltage violation rate is 10.29%.

The average branch power violation rate is 0.0%.

The voltage violates 37 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)



### Dialogue 5, LLM Agent k = 1

```

```python
def calculate_voltage_penalty(self, voltage):
    k_v = 500 # Increased scaling factor
    if 0.95 <= voltage <= 1.05:
        voltage_penalty = 0
    elif voltage > 1.05:
        voltage_penalty = k_v * (voltage - 1.05)**2
    else: # voltage < 0.95
        voltage_penalty = k_v * (0.95 - voltage)**2
    return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
    k_p = 100 # Keep the same scaling factor
    if branch_power <= 5.0:
        branch_power_penalty = 0
    else: # branch_power > 5.0
        branch_power_penalty = k_p * (branch_power - 5.0)**2
    return branch_power_penalty
```

```

### Dialogue 5, ADN Operator, k = 2

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

#### # Training Results

The training lasts 500 episodes, the reward converges to -29.95.

#### # Test Results

The average operational cost is 14.83\$.

The average voltage violation rate is 7.134%.

The average branch power violation rate is 0.0%.

The voltage violates 38 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Dialogue 5, LLM Agent k = 2

```

```python
def calculate_voltage_penalty(self, voltage):
    k_v = 1000 # Increased scaling factor
    if 0.95 <= voltage <= 1.05:
        voltage_penalty = 0
    elif voltage > 1.05:
        voltage_penalty = k_v * (voltage - 1.05)**2 + 1000 * (voltage - 1.05)
    else: # voltage < 0.95
        voltage_penalty = k_v * (0.95 - voltage)**2 + 1000 * (0.95 - voltage)
    return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
    k_p = 100 # Keep the same scaling factor
    if branch_power <= 5.0:
        branch_power_penalty = 0
    else: # branch_power > 5.0
        branch_power_penalty = k_p * (branch_power - 5.0)**2 + 100 * (branch_power - 5.0)
    return branch_power_penalty
```

```

### End: Dialogue 5, ADN Operator, k = 3

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is good.

#### # Training Results

The training lasts 500 episodes, the reward converges to -30.55.

#### # Test Results

The average operational cost is 15.25\$.

The average voltage violation rate is 0.03157%.

The average branch power violation rate is 0.0%.

The voltage violates 1 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

## IV. DIALOGUE PROCESS OF 69-BUS CASE

**Begin: Dialogue 1, LLM Agent k = 0**

```

```python
def calculate_voltage_penalty(self, voltage):
    k_v = 100 # Penalty coefficient
    if voltage > 1.05:
        voltage_penalty = k_v * (voltage - 1.05)**2
    elif voltage < 0.95:
        voltage_penalty = k_v * (0.95 - voltage)**2
    else:
        voltage_penalty = 0
    return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
    k_p = 100 # Penalty coefficient
    if branch_power > 4.7:
        branch_power_penalty = k_p * (branch_power - 4.7)**2
    else:
        branch_power_penalty = 0
    return branch_power_penalty
```

```

**Dialogue 1, ADN Operator, k = 1**

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

## # Training Results

The training lasts 500 episodes, the reward converges to -27.66.

## # Test Results

The average operational cost is 27.55\$.

The average voltage violation rate is 4.529%.

The average branch power violation rate is 0.0%.

The voltage violates 43 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Dialogue 1, LLM Agent k = 1

```

```python
def calculate_voltage_penalty(self, voltage):
    k_v = 500 # Increased penalty coefficient
    if voltage > 1.05:
        voltage_penalty = k_v * (voltage - 1.05)**3 # Higher exponent for faster growth
    elif voltage < 0.95:
        voltage_penalty = k_v * (0.95 - voltage)**3 # Higher exponent for faster growth
    else:
        voltage_penalty = 0
    return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
    k_p = 100 # Keep the same penalty coefficient
    if branch_power > 4.7:
        branch_power_penalty = k_p * (branch_power - 4.7)**2
    else:
        branch_power_penalty = 0
    return branch_power_penalty
```

```

### Dialogue 1, ADN Operator, k = 2

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

#### # Training Results

The training lasts 500 episodes, the reward converges to -25.64.

#### # Test Results

The average operational cost is 25.63\$.

The average voltage violation rate is 4.469%.

The average branch power violation rate is 0.0%.

The voltage violates 43 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Dialogue 1, LLM Agent k = 2

```

```python
def calculate_voltage_penalty(self, voltage):
    k_v = 1000 # Increased penalty coefficient
    if voltage > 1.05:
        voltage_penalty = k_v * ((voltage - 1.05)**3 + (voltage - 1.05)) # Higher exponent and linear term
    elif voltage < 0.95:
        voltage_penalty = k_v * ((0.95 - voltage)**3 + (0.95 - voltage)) # Higher exponent and linear term
    else:
        voltage_penalty = 0
    return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
    k_p = 100 # Keep the same penalty coefficient
    if branch_power > 4.7:
        branch_power_penalty = k_p * (branch_power - 4.7)**2
    else:
        branch_power_penalty = 0
    return branch_power_penalty
```

```

### End: Dialogue 1, ADN Operator, k = 3

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)  
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is good.

#### # Training Results

The training lasts 500 episodes, the reward converges to -31.58.

#### # Test Results

The average operational cost is 31.42\$.

The average voltage violation rate is 0.06039%.

The average branch power violation rate is 0.0%.

The voltage violates 2 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

### Begin: Dialogue 2, LLM Agent k = 0

```

```python
def calculate_voltage_penalty(self, voltage):
    lower_limit = 0.95
    upper_limit = 1.05

    if lower_limit <= voltage <= upper_limit:
        return 0 # No penalty if within the safe range
    else:
        # Quadratic penalty for deviations
        if voltage < lower_limit:
            penalty = (lower_limit - voltage) ** 2
        else:
            penalty = (voltage - upper_limit) ** 2

        # Normalize the penalty to a reasonable scale
        penalty = penalty * 100 # Scale factor to balance with operational cost
        return penalty

def calculate_branch_power_penalty(self, branch_power):
    capacity = 4.7 # in MVA

    if branch_power <= capacity:
        return 0 # No penalty if within the safe range
    else:
        # Quadratic penalty for deviations
        penalty = (branch_power - capacity) ** 2

        # Normalize the penalty to a reasonable scale
        penalty = penalty * 100 # Scale factor to balance with operational cost
        return penalty
...

```

Dialogue 2, ADN Operator, k = 1

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

Training Results

The training lasts 500 episodes, the reward converges to -26.03.

Test Results

The average operational cost is 25.92\$.

The average voltage violation rate is 4.484%.

The average branch power violation rate is 0.0%.

The voltage violates 43 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

Dialogue 2, LLM Agent k = 1

```

'''python
def calculate_voltage_penalty(self, voltage):
    lower_limit = 0.95
    upper_limit = 1.05

    if lower_limit <= voltage <= upper_limit:
        return 0 # No penalty if within the safe range
    else:
        # Quadratic penalty for deviations
        if voltage < lower_limit:
            penalty = (lower_limit - voltage) ** 2
        else:
            penalty = (voltage - upper_limit) ** 2

        # Increase the scaling factor to make the penalty more stringent
        penalty = penalty * 500 # Increased scale factor
        return penalty

def calculate_branch_power_penalty(self, branch_power):
    capacity = 4.7 # in MVA

    if branch_power <= capacity:
        return 0 # No penalty if within the safe range
    else:
        # Quadratic penalty for deviations
        penalty = (branch_power - capacity) ** 2

        # Slightly reduce the scaling factor to balance with operational cost
        penalty = penalty * 50 # Reduced scale factor
        return penalty
'''

```

Dialogue 2, ADN Operator, k = 2

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

Training Results

The training lasts 500 episodes, the reward converges to -26.19.

Test Results

The average operational cost is 25.72\$.

The average voltage violation rate is 4.423%.

The average branch power violation rate is 0.0%.

The voltage violates 43 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

Dialogue 2, LLM Agent k = 2

```

'''python
def calculate_voltage_penalty(self, voltage):
    lower_limit = 0.95
    upper_limit = 1.05

    if lower_limit <= voltage <= upper_limit:
        return 0 # No penalty if within the safe range
    else:
        # Cubic penalty for deviations
        if voltage < lower_limit:
            penalty = (lower_limit - voltage) ** 3
        else:
            penalty = (voltage - upper_limit) ** 3

        # Increase the scaling factor to make the penalty more stringent
        penalty = penalty * 1000 # Increased scale factor
        return penalty

def calculate_branch_power_penalty(self, branch_power):
    capacity = 4.7 # in MVA

    if branch_power <= capacity:
        return 0 # No penalty if within the safe range
    else:
        # Quadratic penalty for deviations
        penalty = (branch_power - capacity) ** 2

        # Slightly reduce the scaling factor to balance with operational cost
        penalty = penalty * 25 # Reduced scale factor
        return penalty
'''

```

Dialogue 2, ADN Operator, k = 3

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

Training Results

The training lasts 500 episodes, the reward converges to -27.21.

Test Results

The average operational cost is 27.18\$.

The average voltage violation rate is 4.589%.

The average branch power violation rate is 0.0%.

The voltage violates 43 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

Dialogue 2, LLM Agent k = 3

```

```python
def calculate_voltage_penalty(self, voltage):
 lower_limit = 0.95
 upper_limit = 1.05

 if lower_limit <= voltage <= upper_limit:
 return 0 # No penalty if within the safe range
 else:
 # Linear and quadratic penalty for deviations
 if voltage < lower_limit:
 deviation = lower_limit - voltage
 penalty = 500 * (deviation + 100 * (deviation ** 2))
 else:
 deviation = voltage - upper_limit
 penalty = 500 * (deviation + 100 * (deviation ** 2))

 return penalty

def calculate_branch_power_penalty(self, branch_power):
 capacity = 4.7 # in MVA

 if branch_power <= capacity:
 return 0 # No penalty if within the safe range
 else:
 # Quadratic penalty for deviations
 deviation = branch_power - capacity
 penalty = 100 * (deviation ** 2)

 return penalty
```

```

End: Dialogue 2, ADN Operator, k = 4

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is good.

Training Results

The training lasts 500 episodes, the reward converges to -30.98.

Test Results

The average operational cost is 30.92\$.

The average voltage violation rate is 0.04529%.

The average branch power violation rate is 0.0%.

The voltage violates 3 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

Begin: Dialogue 3, LLM Agent k = 0

```

```python
def calculate_voltage_penalty(self, voltage):
 # Define the upper and lower limits
 upper_limit = 1.05
 lower_limit = 0.95

 # Calculate the deviation from the limits
 if voltage > upper_limit:
 deviation = voltage - upper_limit
 elif voltage < lower_limit:
 deviation = lower_limit - voltage
 else:
 return 0.0 # No penalty if within the safe range

 # Quadratic penalty
 penalty = deviation ** 2

 return penalty

def calculate_branch_power_penalty(self, branch_power):
 # Define the capacity limit
 capacity_limit = 4.7 # in MVA

 # Calculate the excess power
 if branch_power > capacity_limit:
 excess_power = branch_power - capacity_limit
 else:
 return 0.0 # No penalty if within the safe range

 # Quadratic penalty
 penalty = excess_power ** 2

 return penalty
```

```

Dialogue 3, ADN Operator, k = 1

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

Training Results

The training lasts 500 episodes, the reward converges to -24.69.

Test Results

The average operational cost is 24.69\$.

The average voltage violation rate is 4.574%.

The average branch power violation rate is 0.0%.

The voltage violates 43 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

Dialogue 3, LLM Agent k = 1

```

```python
def calculate_voltage_penalty(self, voltage):
 # Define the upper and lower limits
 upper_limit = 1.05
 lower_limit = 0.95

 # Calculate the deviation from the limits
 if voltage > upper_limit:
 deviation = voltage - upper_limit
 elif voltage < lower_limit:
 deviation = lower_limit - voltage
 else:
 return 0.0 # No penalty if within the safe range

 # Cubic penalty with a scaling factor
 scaling_factor = 1000 # Adjust this factor to balance safety and operational cost
 penalty = scaling_factor * (deviation ** 3)

 return penalty

def calculate_branch_power_penalty(self, branch_power):
 # Define the capacity limit
 capacity_limit = 4.7 # in MVA

 # Calculate the excess power
 if branch_power > capacity_limit:
 excess_power = branch_power - capacity_limit
 else:
 return 0.0 # No penalty if within the safe range

 # Quadratic penalty
 penalty = excess_power ** 2

 return penalty
```

```

Dialogue 3, ADN Operator, k = 2

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)
The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

Training Results

The training lasts 500 episodes, the reward converges to -24.83.

Test Results

The average operational cost is 24.81\$.

The average voltage violation rate is 4.604%.

The average branch power violation rate is 0.0%.

The voltage violates 44 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

Dialogue 3, LLM Agent k = 2

```

```python
def calculate_voltage_penalty(self, voltage):
 # Define the upper and lower limits
 upper_limit = 1.05
 lower_limit = 0.95

 # Calculate the deviation from the limits
 if voltage > upper_limit:
 deviation = voltage - upper_limit
 elif voltage < lower_limit:
 deviation = lower_limit - voltage
 else:
 return 0.0 # No penalty if within the safe range

 # Quartic penalty with a larger scaling factor
 scaling_factor = 10000 # Adjust this factor to balance safety and operational cost
 penalty = scaling_factor * (deviation ** 4)

 return penalty

def calculate_branch_power_penalty(self, branch_power):
 # Define the capacity limit
 capacity_limit = 4.7 # in MVA

 # Calculate the excess power
 if branch_power > capacity_limit:
 excess_power = branch_power - capacity_limit
 else:
 return 0.0 # No penalty if within the safe range

 # Quadratic penalty with a scaling factor
 scaling_factor = 100 # Adjust this factor to balance safety and operational cost
 penalty = scaling_factor * (excess_power ** 2)

 return penalty
```

```

Dialogue 3, ADN Operator, k = 3

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

Training Results

The training lasts 500 episodes, the reward converges to -25.83.

Test Results

The average operational cost is 25.82\$.

The average voltage violation rate is 4.574%.

The average branch power violation rate is 0.0%.

The voltage violates 43 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

Dialogue 3, LLM Agent k = 3

```

```python
def calculate_voltage_penalty(self, voltage):
 # Define the upper and lower limits
 upper_limit = 1.05
 lower_limit = 0.95

 # Calculate the deviation from the limits
 if voltage > upper_limit:
 deviation = voltage - upper_limit
 elif voltage < lower_limit:
 deviation = lower_limit - voltage
 else:
 return 0.0 # No penalty if within the safe range

 # Linear and quadratic penalty with a scaling factor
 linear_scaling_factor = 1000 # Adjust this factor to balance safety and operational cost
 quadratic_scaling_factor = 10000 # Adjust this factor to balance safety and operational cost

 # Combined penalty
 penalty = linear_scaling_factor * deviation + quadratic_scaling_factor * (deviation ** 2)

 return penalty

def calculate_branch_power_penalty(self, branch_power):
 # Define the capacity limit
 capacity_limit = 4.7 # in MVA

 # Calculate the excess power
 if branch_power > capacity_limit:
 excess_power = branch_power - capacity_limit
 else:
 return 0.0 # No penalty if within the safe range

 # Quadratic penalty with a smaller scaling factor
 scaling_factor = 10 # Adjust this factor to balance safety and operational cost
 penalty = scaling_factor * (excess_power ** 2)

 return penalty
```

```

End: Dialogue 3, ADN Operator, k = 4

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)
The performance of the RL agent with the designed voltage penalty function and branch power penalty function is good.

Training Results

The training lasts 500 episodes, the reward converges to -30.78.

Test Results

The average operational cost is 30.69\$.

The average voltage violation rate is 0.03019%.

The average branch power violation rate is 0.0%.

The voltage violates 2 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

Begin: Dialogue 4, LLM Agent k = 0

```

```python
def calculate_voltage_penalty(self, voltage):
 k_v = 100 # Scaling factor for voltage penalty
 if voltage > 1.05:
 voltage_penalty = k_v * (voltage - 1.05)**2
 elif voltage < 0.95:
 voltage_penalty = k_v * (0.95 - voltage)**2
 else:
 voltage_penalty = 0
 return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
 k_p = 100 # Scaling factor for branch power penalty
 if branch_power > 4.7:
 branch_power_penalty = k_p * (branch_power - 4.7)**2
 else:
 branch_power_penalty = 0
 return branch_power_penalty
```

```

Dialogue 4, ADN Operator, k = 1

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

Training Results

The training lasts 500 episodes, the reward converges to -25.76.

Test Results

The average operational cost is 25.66\$.

The average voltage violation rate is 4.514%.

The average branch power violation rate is 0.0%.

The voltage violates 43 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

Dialogue 4, LLM Agent k = 1

```

```python
def calculate_voltage_penalty(self, voltage):
 k_v_linear = 1000 # Linear scaling factor for voltage penalty
 k_v_quadratic = 100 # Quadratic scaling factor for voltage penalty

 if voltage > 1.05:
 voltage_penalty = k_v_linear * (voltage - 1.05) + k_v_quadratic * (voltage - 1.05)**2
 elif voltage < 0.95:
 voltage_penalty = k_v_linear * (0.95 - voltage) + k_v_quadratic * (0.95 - voltage)**2
 else:
 voltage_penalty = 0
 return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
 k_p = 50 # Slightly reduced scaling factor for branch power penalty
 if branch_power > 4.7:
 branch_power_penalty = k_p * (branch_power - 4.7)**2
 else:
 branch_power_penalty = 0
 return branch_power_penalty
```

```

Dialogue 4, ADN Operator, k = 2

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is good.

Training Results

The training lasts 500 episodes, the reward converges to -31.22.

Test Results

The average operational cost is 31.13\$.

The average voltage violation rate is 0.03019%.

The average branch power violation rate is 0.0%.

The voltage violates 2 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

Dialogue 4, LLM Agent k = 2

```

python
def calculate_voltage_penalty(self, voltage):
    k_v_linear = 800 # Reduced linear scaling factor for voltage penalty
    k_v_quadratic = 80 # Reduced quadratic scaling factor for voltage penalty

    if voltage > 1.05:
        voltage_penalty = k_v_linear * (voltage - 1.05) + k_v_quadratic * (voltage - 1.05)**2
    elif voltage < 0.95:
        voltage_penalty = k_v_linear * (0.95 - voltage) + k_v_quadratic * (0.95 - voltage)**2
    else:
        voltage_penalty = 0
    return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
    k_p = 40 # Further reduced scaling factor for branch power penalty
    if branch_power > 4.7:
        branch_power_penalty = k_p * (branch_power - 4.7)**2
    else:
        branch_power_penalty = 0
    return branch_power_penalty

```

End: Dialogue 4, ADN Operator, k = 3

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is good.

Training Results

The training lasts 500 episodes, the reward converges to -30.70.

Test Results

The average operational cost is 30.67\$.

The average voltage violation rate is 0.04529%.

The average branch power violation rate is 0.0%.

The voltage violates 3 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

Begin: Dialogue 5, LLM Agent k = 0

```

```python
def calculate_voltage_penalty(self, voltage):
 # Define the voltage limits
 upper_limit = 1.05
 lower_limit = 0.95

 # Calculate the deviation from the safe range
 if voltage > upper_limit:
 deviation = voltage - upper_limit
 elif voltage < lower_limit:
 deviation = lower_limit - voltage
 else:
 deviation = 0.0

 # Quadratic penalty for the deviation
 voltage_penalty = deviation ** 2

 return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
 # Define the branch power capacity
 capacity = 4.7 # in MVA

 # Calculate the deviation from the safe range
 if branch_power > capacity:
 deviation = branch_power - capacity
 else:
 deviation = 0.0

 # Quadratic penalty for the deviation
 branch_power_penalty = deviation ** 2

 return branch_power_penalty
```

```

Dialogue 5, ADN Operator, k = 1

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

Training Results

The training lasts 500 episodes, the reward converges to -24.85.

Test Results

The average operational cost is 24.84\$.

The average voltage violation rate is 4.544%.

The average branch power violation rate is 0.0%.

The voltage violates 43 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

Dialogue 5, LLM Agent k = 1

```

```python
def calculate_voltage_penalty(self, voltage):
 # Define the voltage limits
 upper_limit = 1.05
 lower_limit = 0.95

 # Calculate the deviation from the safe range
 if voltage > upper_limit:
 deviation = voltage - upper_limit
 elif voltage < lower_limit:
 deviation = lower_limit - voltage
 else:
 deviation = 0.0

 # Cubic penalty for the deviation with a scaling factor
 voltage_penalty = (deviation ** 3) * 1000 # Increase the scaling factor to 1000

 return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
 # Define the branch power capacity
 capacity = 4.7 # in MVA

 # Calculate the deviation from the safe range
 if branch_power > capacity:
 deviation = branch_power - capacity
 else:
 deviation = 0.0

 # Quadratic penalty for the deviation with a smaller scaling factor
 branch_power_penalty = (deviation ** 2) * 100 # Decrease the scaling factor to 100

 return branch_power_penalty
```

```

Dialogue 5, ADN Operator, k = 2

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

Training Results

The training lasts 500 episodes, the reward converges to -27.17.

Test Results

The average operational cost is 27.14\$.

The average voltage violation rate is 4.635%.

The average branch power violation rate is 0.0%.

The voltage violates 44 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

Dialogue 5, LLM Agent k = 2

```

```python
def calculate_voltage_penalty(self, voltage):
 # Define the voltage limits
 upper_limit = 1.05
 lower_limit = 0.95

 # Calculate the deviation from the safe range
 if voltage > upper_limit:
 deviation = voltage - upper_limit
 elif voltage < lower_limit:
 deviation = lower_limit - voltage
 else:
 deviation = 0.0

 # Cubic penalty for the deviation with a larger scaling factor
 voltage_penalty = (deviation ** 3) * 5000 # Increase the scaling factor to 5000

 return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
 # Define the branch power capacity
 capacity = 4.7 # in MVA

 # Calculate the deviation from the safe range
 if branch_power > capacity:
 deviation = branch_power - capacity
 else:
 deviation = 0.0

 # Quadratic penalty for the deviation with a smaller scaling factor
 branch_power_penalty = (deviation ** 2) * 10 # Decrease the scaling factor to 10

 return branch_power_penalty
```

```

Dialogue 5, ADN Operator, k = 3

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)
The performance of the RL agent with the designed voltage penalty function and branch power penalty function is bad.

Training Results

The training lasts 500 episodes, the reward converges to -25.34.

Test Results

The average operational cost is 25.22\$.

The average voltage violation rate is 4.484%.

The average branch power violation rate is 0.0%.

The voltage violates 43 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)

Dialogue 5, LLM Agent k = 3

```

```python
def calculate_voltage_penalty(self, voltage):
 # Define the voltage limits
 upper_limit = 1.05
 lower_limit = 0.95

 # Calculate the deviation from the safe range
 if voltage > upper_limit:
 deviation = voltage - upper_limit
 elif voltage < lower_limit:
 deviation = lower_limit - voltage
 else:
 deviation = 0.0

 # Combined linear and cubic penalty for the deviation with a larger scaling factor
 voltage_penalty = (deviation * 1000) + (deviation ** 3) * 10000 # Increase the scaling factors

 return voltage_penalty

def calculate_branch_power_penalty(self, branch_power):
 # Define the branch power capacity
 capacity = 4.7 # in MVA

 # Calculate the deviation from the safe range
 if branch_power > capacity:
 deviation = branch_power - capacity
 else:
 deviation = 0.0

 # Quadratic penalty for the deviation with a smaller scaling factor
 branch_power_penalty = (deviation ** 2) * 1 # Further decrease the scaling factor to 1

 return branch_power_penalty
```

```

End: Dialogue 5, ADN Operator, k = 4

The designed penalty functions are: (Designed function is the LLM agent output, we omit here considering the length)
 The performance of the RL agent with the designed voltage penalty function and branch power penalty function is good.

Training Results

The training lasts 500 episodes, the reward converges to -30.80.

Test Results

The average operational cost is 30.69\$.

The average voltage violation rate is 0.03019%.

The average branch power violation rate is 0.0%.

The voltage violates 2 times in one day.

The branch power violates 0 times in one day. (CoT suggestions are also omitted here, which have been shown in Section II)