Supplementary Files for LLM Powered Automated Modeling and Optimization of Active Distribution Network Dispatch Problems

I. Configurations of Test Cases

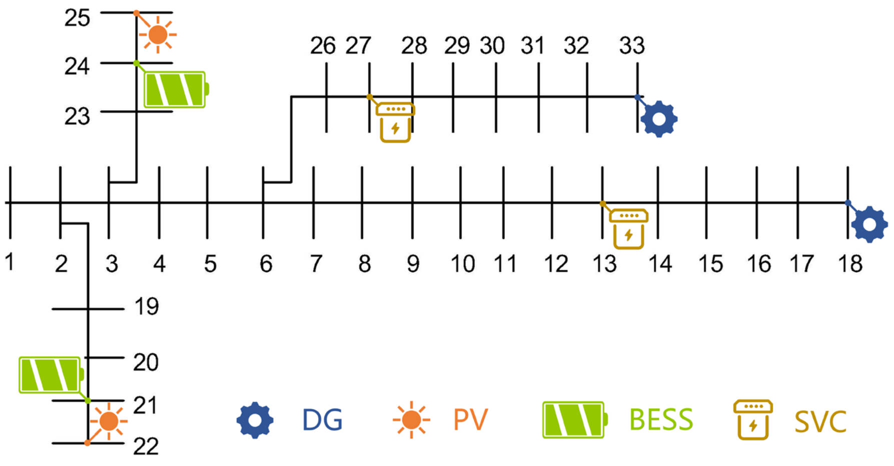

Fig. 1. Topology of 33-bus system.

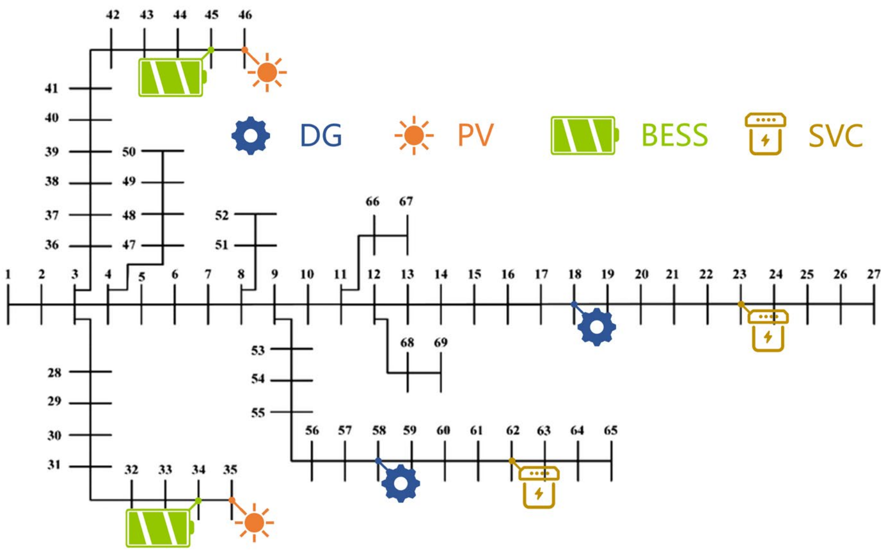

Fig. 2. Topology of 69-bus system.

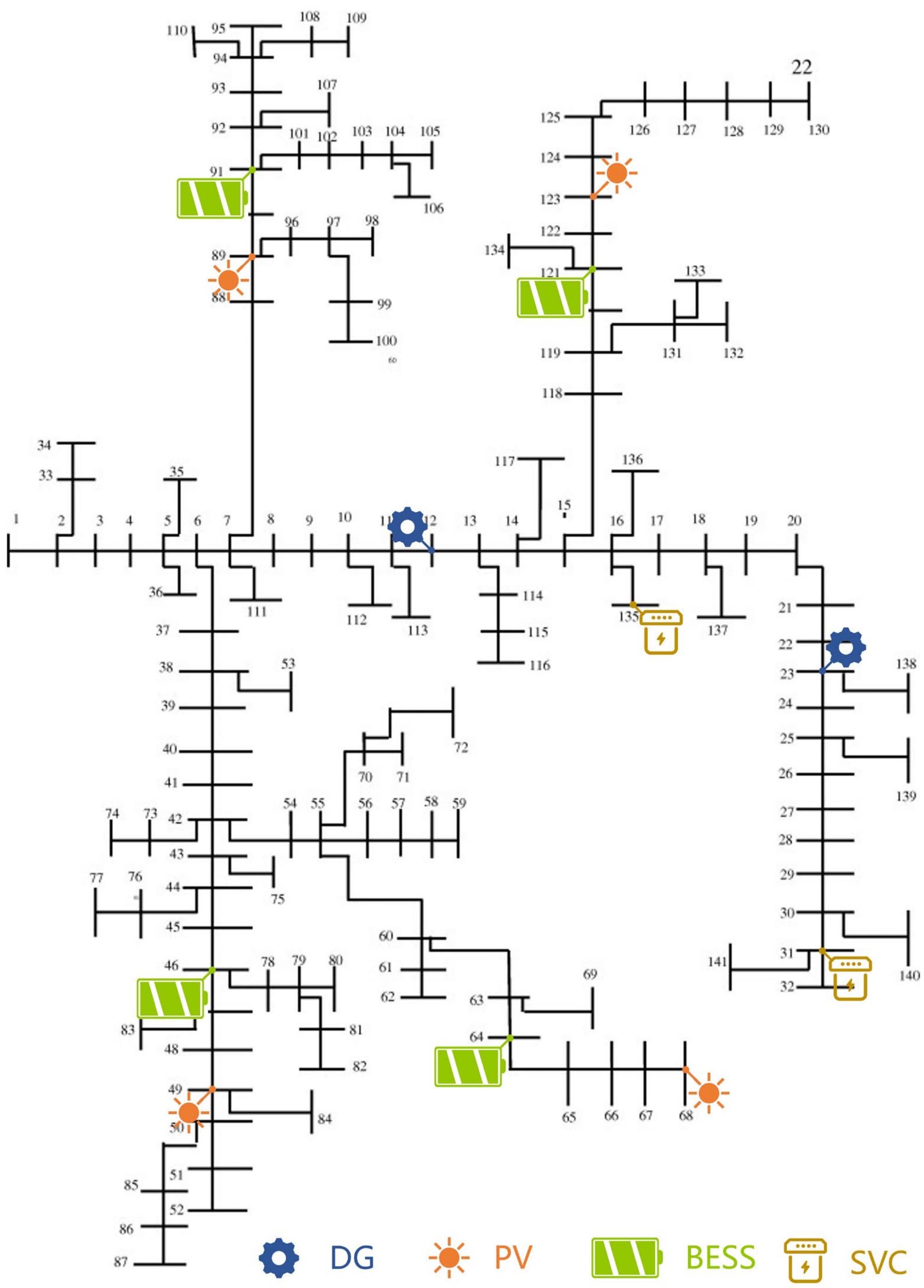

Fig. 3. Topology of 141-bus system.

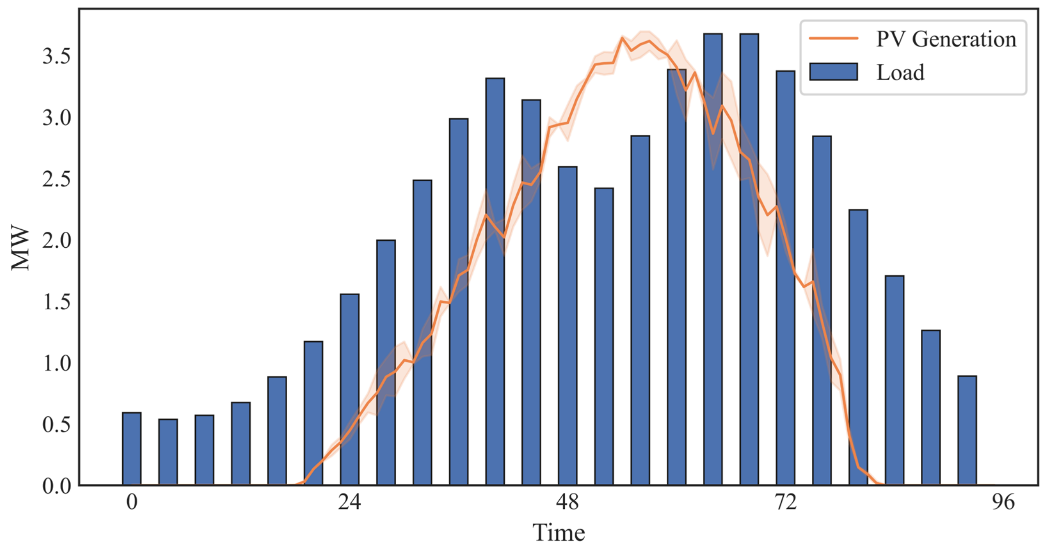

Fig. 4. PV generation and load profiles of 33-bus system.

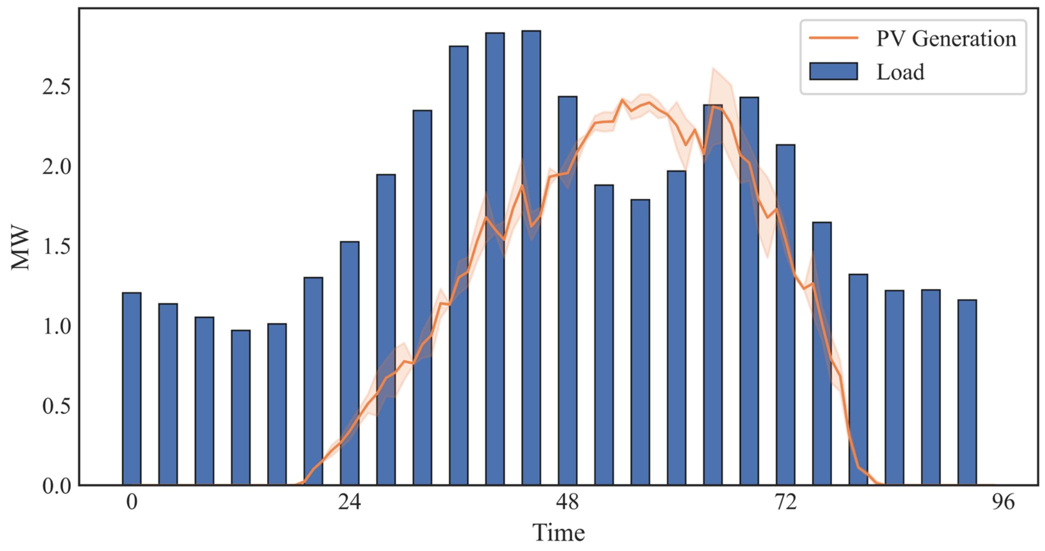

Fig. 5. PV generation and load profiles of 69-bus system.

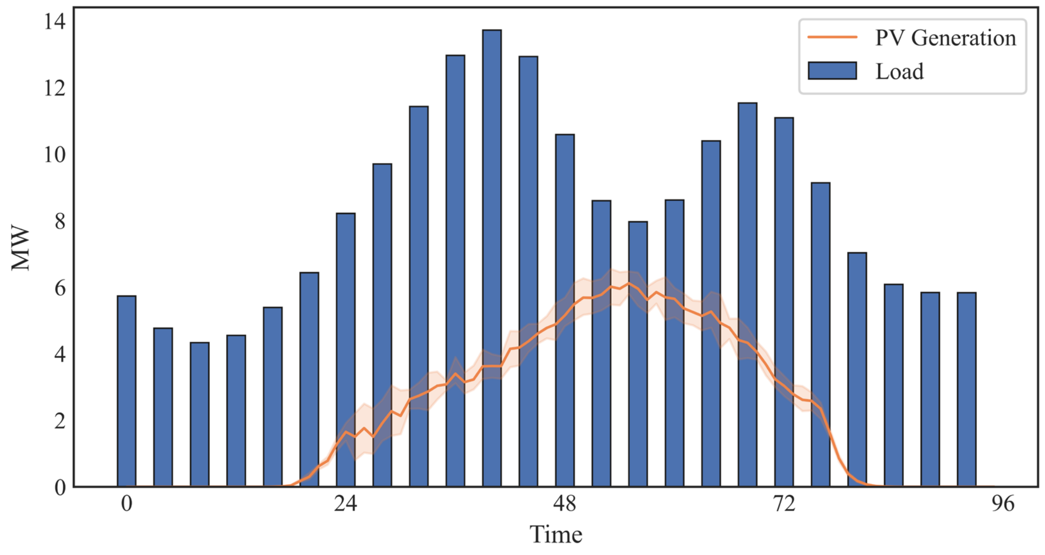

Fig. 6. PV generation and load profiles of 141-bus system.

TABLE I
PARAMETERS OF TEST SYSTEMS

Parameters	33-bus	69-bus	141-bus
Number of DGs	2	2	2
Number of PVs	2	2	4
Number of BESSs	2	2	4
Number of SVCs	2	2	2
Control interval (minute)	15	15	15
Voltage limitations (p.u.)	[0.95, 1.05]	[0.95, 1.05]	[0.95, 1.05]
Branch capacity (MVA)	4.0	3.6	15.0
ρ^{DG}	0.3	0.3	0.3
$\rho^{BESS,dis}$	0.1	0.1	0.1
$\rho^{BESS,cha}$	0.1	0.1	0.1
ρ_0	0.5	0.5	0.5
$P_{min}^{DG}, P_{max}^{DG}$ (MW)	0.0, 1.5	0.0, 1.5	0.0, 6.0
$Q_{min}^{DG}, Q_{max}^{DG}$ (MVAR)	-0.45, 0.45	-0.45, 0.45	-1.0, 1.0
R_{max}^{DG} (MW/h)	1.6	1.6	3.2
S_{max}^{PV} (MVA)	5.4	4.6	11.0
$P_{max}^{BESS,dis}$ (MW)	1.0	1.0	2.0
$P_{max}^{BESS,cha}$ (MW)	1.0	1.0	2.0
Q_{max}^{SVC}(MVAR)	0.4	0.4	0.8
η	0.95	0.95	0.95

II. Prompts for Information Extractor

You are an expert in power system operation and optimization. The user has a distribution network and will give a dispatch request, you need to identify and extract useful and relevant information from this request.

Here is the information about the distribution network:

# District
This distribution network has three districts: Valley District, Railway District, and Business District. The request will only specify one District from these three Districts.

# Objective
The user has two types of objectives: Day-ahead objective and Single-timestep objective.
Day-ahead objective includes: Minimize cost, Minimize power loss, and Minimize voltage deviation. Single-timestep objective includes: Eliminate voltage violation, and Eliminate branch power violation. The request will only include one Objective from these Objectives.

# Equipment
This distribution network has the following equipment: diesel generator (DG), battery energy storage system (BESS), photovoltaic (PV), and static VAR compensator (SVC). The request may include several available Equipment from these Equipment.

# Constraint
There may be some additional constraints in the request: No voltage violation, No branch power violation, additional Generation constraints for the equipment, State of charge (SOC) constraint for the BESS. The request may include several additional Constraints from these Constraints.

The dispatch request may explicitly or implicitly include the District, Objective, available Equipment, and additional Constraint information. Your task is to identify and extract them into the required output format:

# Output format
<District> One district you identified from Valley District, Railway District, and Business District </District>
<Objective> One objective you identified from Minimize cost, Minimize power loss, Minimize voltage deviation, Eliminate voltage violation, and Eliminate branch power violation </Objective>
<Equipment> All equipment you identified from DG, BESS, PV, SVC </Equipment>
<Constraint> All additional constraint you identified, such as No voltage violation, No branch power violation, SOC constraint, Generation constraint </Constraint>

Here are some examples:

# Example 1
Request:
Please minimize the operational cost of Railway District for tomorrow. You can utilize resources such as diesel generators, energy storage, and photovoltaic systems. Note: Ensure that the voltage remains within limits and the lines are not overloaded.
Output:
<District> Railway District </District>
<Objective> Minimize cost </Objective>
<Equipment> DG, BESS, PV </Equipment>
<Constraint> No voltage violation, No branch power violation </Constraint> (next page)

# Example 2
Request:
Tomorrow, the upper grid will inspect the Valley District, and we need to minimize the voltage deviations. All resources can be dispatched. Additionally, it is recommended to keep the SOC of the BESS within the 40% to 80% range.
Output:
<District> Valley District </District>
<Objective> Minimize voltage deviation </Objective>
<Equipment> DG, BESS, PV, SVC </Equipment>
<Constraint> SOC of BESS in 40%~80% </Constraint>

# Example 3
Request:
Attention: voltage in Business District has fallen below the lower limit. Please utilize PV and SVC to ensure voltage safety.
Output:
<District> Business District </District>
<Objective> Eliminate voltage violation </Objective>
<Equipment> PV, SVC </Equipment>
<Constraint> None </Constraint>

Here is the dispatch request, please based on the given information and examples, identify and extract District, Objective, available Equipment, and additional Constraint. Let's think step by step!

III. Prompts for Problem Formulator

System Prompts

You are an expert in power system operation and optimization. The user has a dispatch request for a distribution network. In the following multi-round dialogues, please sequentially write the required information in math format according to the given explanations and examples.

Dlg. 1: Objective Function

The objective of the user is <Objective>.

The available equipment includes <Equipment>.

This is the detailed explanation of this objective:
<Objective_Explanation>

According to the given objective, available equipment, explanation, and examples, please write the objective function in math format. Remember: You only need to write the objective function, do not write other math equations. Also, you must strictly follow the explanation and example, do not write the objective function on your own.

Dlg. 2: Equipment Constraints

The objective of the user is <Objective>.

The available equipment includes <Equipment>.

This is the detailed explanation of equipment constraints:
<Equipment_Constraint_Explanation>

According to the given objective, available equipment, explanation, and examples, please write all of the relevant equipment constraints in math format. Remember: You only need to write the equipment constraints, do not write other math equations.

Dlg. 3: Power Flow Constraints

The available equipment includes <Equipment>.

This optimization problem also needs to satisfy power flow constraints, which include Power injection constraint, Distflow constraint, and Root bus voltage constraint.

# Power injection constraint
This constraint describes how the active and reactive power injection at each bus are related to power generation of available equipment. You can utilize \(P_{i, t} \) and \(Q_{i, t} \) to express the active and reactive power injection at bus-i, time-t, \(P^{DG}_{i, t} \) and \(Q^{DG}_{i, t} \) to express the active and reactive power generation of DG at bus-i, time-t, \(P^{BESS, dis}_{i, t} \) and \(P^{BESS, cha}_{i, t} \) to express the discharging and charging power of BESS at bus-i, time-t, \(P^{PV}_{i, t} \) and \(Q^{PV}_{i, t} \) to express the active and reactive power generation of PV at bus-i, time-t, \(Q^{SVC}_{i, t} \) to express the reactive power generation of SVC at bus-i, time-t. \(P^{load}_{i, t} \) and \(Q^{load}_{i, t} \) are the active and reactive load at bus-i, time-t.

Here are some examples:

# Example 1
Available equipment:
DG, BESS, PV, SVC

Math format:
\(P_{i, t} = P^{DG}_{i, t} + P^{BESS, dis}_{i, t} - P^{BESS, cha}_{i, t} + P^{PV}_{i, t} - P^{load}_{i, t} \quad \forall i, t \)
\(Q_{i, t} = Q^{DG}_{i, t} + Q^{PV}_{i, t} + Q^{SVC}_{i, t} - Q^{load}_{i, t} \quad \forall i, t \)

# Example 2
Available equipment:
DG, BESS

Math format:
\(P_{i, t} = P^{DG}_{i, t} + P^{BESS, dis}_{i, t} - P^{BESS, cha}_{i, t} - P^{load}_{i, t} \quad \forall i, t \)
\(Q_{i, t} = Q^{DG}_{i, t} - Q^{load}_{i, t} \quad \forall i, t \) (next page)

```
# Example 3
Available equipment:
PV, SVC

Math format:
\( P_{i, t} = P^{PV}_{i, t} - P^{load}_{i, t} \quad \forall i, t \)
\( Q_{i, t} = Q^{PV}_{i, t} + Q^{SVC}_{i, t} - Q^{load}_{i, t} \quad \forall i, t \)

# Distflow constraint
This constraint describes the power flow relationship in the distribution network. You can utilize \( P_{ij, t} \) and \( Q_{ij, t} \) to express the acitve and reactive power flow in branch-ij, time-t, \( r_{ij} \) and \( x_{ij} \) to express the resistance and reactance of branch-ij, \( l_{ij, t} \) is the squared current flow in branch-ij, time-t, \( v_{i, t} \) is the squared voltage magnitude at bus-i, time-t. Then the Distflow constraint in math format is:
\( P_{j, t} = \sum_{k: j \rightarrow k} P_{jk, t} - \sum_{i: i \rightarrow j} (P_{ij, t} - r_{ij} l_{ij, t}) \quad \forall j, t \)
\( Q_{j, t} = \sum_{k: j \rightarrow k} Q_{jk, t} - \sum_{i: i \rightarrow j} (Q_{ij, t} - x_{ij} l_{ij, t}) \quad \forall j, t \)
\( v_{j, t} = v_{i, t} - 2 (r_{ij} P_{ij, t} + x_{ij} Q_{ij, t}) + (r_{ij}^2 + x_{ij}^2) l_{ij, t} \quad \forall ij, t \)
\( \begin{Vmatrix} 2P_{ij, t} \\ 2Q_{ij, t} \\ l_{ij, t} - v_{i, t} \end{Vmatrix}_2 \leq l_{ij, t} + v_{i, t} \quad \forall ij, t \)

# Root bus voltage constraint
This constraint describes the root bus voltage must be 1.0 p.u.. You can utilize \( v_{0, t} \) to express the squared root bus voltage magnitude at time-t. Then root bus voltage constraint in math format is:
\( v_{0, t} = 1.0 \quad \forall t \)

According to the given available equipment, explanation, and examples, please write the power flow constraints (including Power injection constraint, Distflow constraint, and Root bus voltage constraint) in math format. Remember: You only need to write the Power injection constraint, Distflow constraint, and Root bus voltage constraint, do not write other math equations.
```

Dlg. 4: Additional Constraints

```
The available equipment includes <Equipment>.

The additional constraints include <Additional_Constraints>.

Here are some typical examples of additional constraints and their math formats:

# Example 1: voltage safety constraint
Additional constraint:
"No voltage violations please." OR "Watch out the bus voltage!" OR "The voltage magnitude of all buses should be within the limits." OR "Attention: Node voltage cannot violate the limit."
Math format:
\( V_{min}^2 \leq v_{i, t} \leq V_{max}^2 \quad \forall i, t \)
Explanation:
\( V_{min} \) is the lower limit of bus voltage, \( V_{max} \) is the upper limit of bus voltage, \( v_{i, t} \) is the squared voltage magnitude at bus-i, time-t. This constraint means the voltage of each node should be greater than the lower limit and less than the upper limit.

# Example 2: branch power safety constraint
Additional constraint:
"No branch power violations please." OR "Watch out the branch power!" OR "The power flow of all branches should be within the limits." OR "Attention: Branch power cannot violate the limit."
Math format:
\( P_{ij, t}^2 + Q_{ij, t}^2 \leq \left( S^{brch}_{max} \right)^2 \quad \forall ij, t \)
Explanation:
\( S^{brch}_{max} \) is the branch power capacity, \( P_{ij, t} \) is the active power flow in branch-ij, time-t, \( Q_{ij, t} \) is the reactive power flow in branch-ij, time-t. This constraint means the power flow in each branch should be less than the capacity. (next page)
```

# Example 3: SOC constraint
Additional constraint:
"SOC of the BESS should be within 30%~80%." OR "I want the SOC of the BESS higher than 0.3 and less than 0.8." OR "Please limit the SOC in [0.3, 0.8]." OR "Attention: SOC should be in the range 30%~80%."
Math format:
\(0.3 \leq SOC^{BESS}_{i, t} \leq 0.8 \quad \forall i, t \)
Explanation:
\(SOC^{BESS}_{i, t} \) is the SOC of the BESS at bus-i, time-t. This constraint means the SOC of all BESSs should be in the range of [0.3, 0.8].

# Example 4: generation constraint
Additional constraint:
"Power generated by DG cannot be higher than 0.7MW." OR "I want the DG less than 0.7MW." OR "Please limit the active DG power lower than 0.7MW." OR "Attention: Remember the DG power less than 0.7MW."
Math format:
\(P^{DG}_{i, t} \leq 0.7 \quad \forall i, t \)
Explanation:
\(P^{DG}_{i, t} \) is the active power generated by DG at bus-i, time-t. This constraint means the power generated by DG is less than 0.7MW.

According to the given available equipment, additional constraints, and examples, please write the additional constraints in math format. Remember: You only need to write the additional constraint, do not write other math equations. If there are no additional constraints, for example, the additional constraint is "None", you just generate "None" and do not generate anything else in this conversation.

Dlg. 5: Complete Problem

The user request is:
<User_Request>

The objective function in math format is:
<Objective_Function>

The equipment constraint in math format is:
<Equipment_Constraint>

The power flow constraint in math format is:
<Power_Flow_Constraint>

The additional constraint in math format is:
<Additional_Constraint>

According to the above information, summarize the complete constrained optimization problem in math format. It should be noted that all equations should adopt a unified definition of symbols. Your output format is:

<Math_Problem>
Please replace this part with the complete constrained optimization problem.
</Math_Problem>

Remember: use <Math_Problem> and </Math_Problem> to wrap the complete constrained optimization problem.

Dlg. 6: Relaxed Problem

```
The user request is:
<User_Request>

The complete constrained optimization problem is:
<Complete_Problem>

There may exist some non-convex objective or constraints in the problem, please relax them.

Here are some examples:

# Example 1: min-max problem
Original equation:
\( \min \max_{i=0}^{N-1} \left( v_{i, t} - 1.0 \right)^2 \)

Relaxed equation:
\( \min z \)
\( z \geq \left( v_{i, t} - 1.0 \right)^2 \quad \forall i \)

Explanation:
\( v_{i, t} \) is the squared voltage magnitued at bus-i, time-t. The objective means minimizing the maximum voltage deviation across all buses so that voltage violation can be eliminated.
To relax it, introduce a new variable \( z \) to represent \( \max_{i=0}^{N-1} \left( v_{i, t} - 1.0 \right)^2 \). Then, introduce a series of inequalities, \( z \geq \left( v_{i, t} - 1.0 \right)^2 \quad \forall i \). As long as \( z \) is larger than voltage deviations across all buses, then minimize it, the final solution of \( z \) is the maximum voltage deviation across all buses.

# Example 2: min-max problem
Original equation:
\( \min \max_{ij} \left( P_{ij, t}^2 + Q_{ij, t}^2 \right) \)

Relaxed equation:
\( \min z \)
\( z \geq \left( P_{ij, t}^2 + Q_{ij, t}^2 \right) \quad \forall ij \)

Explanation:
\( P_{ij, t} \) and \( Q_{ij, t} \) are the active power and reactive power flow in branch-ij, time-t. The objective means minimizing the maximum power in all branches so that branch power violation can be eliminated.
To relax it, introduce a new variable \( z \) to represent \( \max_{ij} \left( P_{ij, t}^2 + Q_{ij, t}^2 \right) \). Then, introduce a series of inequalities, \( z \geq \left( P_{ij, t}^2 + Q_{ij, t}^2 \right) \quad \forall ij \). As long as \( z \) is larger than branch power across all branches, then minimize it, the final solution of \( z \) is the maximum branch power across all branches.

# Example 3: discharge/charge stage constraint
Original equation:
\( P^{BESS, dis}_{i, t} \times P^{BESS, cha}_{i, t} = 0 \quad \forall i, t \)
\( 0 \leq P^{BESS, dis}_{i, t} \leq P^{BESS, dis}_{max} \quad \forall i, t \)
\( 0 \leq P^{BESS, cha}_{i, t} \leq P^{BESS, cha}_{max} \quad \forall i, t \)

Relaxed equation:
\( 0 \leq P^{BESS, dis}_{i, t} \leq P^{BESS, dis}_{max} \times u^{BESS}_{i, t} \quad \forall i, t \)
\( 0 \leq P^{BESS, cha}_{i, t} \leq P^{BESS, cha}_{max} \times (1 - u^{BESS}_{i, t}) \quad \forall i, t \)
\( u^{BESS}_{i, t} \in \{0, 1\} \quad \forall i, t \) (next page)
```

Explanation:
\(P^{BESS, dis}_{i, t} \) and \(P^{BESS, cha}_{i, t} \) are the discharging and charging power of BESS at bus-i, time-t, \(P^{BESS, dis}_{max} \) and \(P^{BESS, cha}_{max} \) are the discharging and charging power upper limit. The BESS cannot discharge and charge simultaneously, so the original equation includes \(P^{BESS, dis}_{i, t} \times P^{BESS, cha}_{i, t} = 0 \quad \forall i, t \).
To relax it, introduce a new binary variable \(u^{BESS}_{i, t} \) indicate the state of BESS at bus-i, time-t. If \(u^{BESS}_{i, t} = 1 \), the BESS is discharging, then the second constraint limits the charging power \(P^{BESS, cha}_{i, t} \) to zero. If \(u^{BESS}_{i, t} = 0 \), the BESS is charging, then the first constraint limits the discharging power \(P^{BESS, dis}_{i, t} \) to zero.

According to the above information and examples, relax the non-convex objectives or constraints in the constrained optimization problem and summarize it in math format. If there does not exists non-convex objectives or constraints in the original problem, do not change anything.
Your output format is:

<Math_Problem>
Please replace this part with the relaxed complete constrained optimization problem
</Math_Problem>

Remember: Only relax the above mentioned min-max objective and discharge/charge state constraint! Quadratic objectives, such as minimize voltage deviation; Quadratic constraints, such as branch power constraint, PV capacity constraint, are all acceptable and do not change them! Do not relax the objective or constraint beyond the above mentioned min-max objective and discharge/charge state constraint!
Also, use <Math_Problem> and </Math_Problem> to wrap the relaxed complete constrained optimization problem.

IV. Prompts for Code Programmer

System Prompts

You are an expert in power system operation, optimization, and programming. The user has a constrained optimization problem in math format and you need to convert the problem into Python code with a new modeling language PyOptInterface. In the following multi-round dialogues, you will be provided 1. common functions in PyOptInterface, 2. parameters of the numerical case, 3. three examples, 4. required Python output format. Please write the complete Python code according to the given explanations and examples.

Case Format Explanations

Here are the explanations about the numerical case:

The numerical case is a Python dictionary, whose keys include:

# parameters
"T": an int, the length of the dispatch duration, which should be considered when creating variables, creating constraints, and when the objective is day-ahead (multi-timestep).
"current_t": an int, current time index, which should be used when the objective is single-timestep.
"BESS_num": an int, number of the BESS.
"MVA": a float, base value of the power, you should divide this number when converting real power value to per-unit value.
"rho_DG": a float, the cost coefficient of DG.
"rho_BESS_dis": a float, the cost coefficient for discharge of BESS.
"rho_BESS_cha": a float, the cost coefficient for charge of BESS.
"rho_0": a float, the electricity price. (next page)

"bus": a numpy array, the model of the bus in the distribution network, you can utilize .shape[0] to calculate the number of buses.
"branch": a numpy array, the model of the branch in the distribution network, you can utilize .shape[0] to calculate the number of branches, utilize [:, 0].astype(int) - 1 to get the starting buses of the branches, utilize [:, 1].astype(int) - 1 to get the end buses of the branches, utilize [:, 2] to get the resistances of the branches, utilize [:, 3] to get the reactances of the branches.
"gen": a numpy array, the model of the devices in the distribution network, you can utilize .shape[0] to calculate the number of devices.
"P_DG_max": a float, the upper limit of the DG active power.
"P_DG_min": a float, the lower limit of the DG active power.
"Q_DG_max": a float, the upper limit of the DG reactive power.
"Q_DG_min": a float, the lower limit of the DG reactive power.
"R_max": a float, the ramping rate limit of DG.
"S_PV_max": a float, the capacity of PV.
"P_BESS_dis_max": a float, the upper limit of the discharging power of the BESS.
"P_BESS_cha_max": a float, the upper limit of the charging power of the BESS.
"SOC_BESS_max": a float, the upper limit of SOC of the BESS.
"SOC_BESS_min": a float, the lower limit of SOC of the BESS.
"SOC_init": a float, the initial SOC of the BESS.
"Q_SVC_max": a float, the upper limit of the SVC reactive power.
"eta": a float, the discharging/charging efficiency of the BESS.
"V_max": a float, the upper limit of bus voltage.
"V_min": a float, the lower limit of bus voltage.
"S_branch_max": a float, the branch power capacity.

# models
"DG_bus_index": a dictionary, which indicates the bus of the DG and the DG index in the devices. the key is the bus, and the value is DG index in the devices.
"PV_bus_index": a dictionary, which indicates the bus of the PV and the PV index in the devices. the key is the bus, and the value is PV index in the devices.
"BESS_bus_index": a dictionary, which indicates the bus of the BESS and the BESS index in the devices. the key is the bus, and the value is BESS index in the devices.
"SVC_bus_index": a dictionary, which indicates the bus of the SVC and the SVC index in the devices. the key is the bus, and the value is SVC index in the devices.

# data
"PV": a T * PV_num numpy array, the PV generation at each timestep.
"P_load": a T * bus_num numpy array, the active power load at each timestep.
"Q_load": a T * bus_num numpy array, the reactive power load at each timestep.

Remember: if PV is available, you should relate the PV generation variable with the PV data, this constraint may not be included in math format, please refer to the examples.

PyOptInterface Explanations

Here are the explanations about the modeling language PyOptInterface:

# Usage
Python code:
```python
import pyoptinterface as poi
from pyoptinterface import gurobi
```
Explanation:
PyOptInterface has no dependencies other than Python itself. You only need to import it. As for the optimizer, we utilize gurobi, and you need to import it from pyoptinterface. (next page)

# Model
Python code:
```python
model = gurobi.Model()
```
Explanation:
A model is a concrete instance tied to a specific solver. To create a model, we need to import the corresponding module and call the constructor of the model class.

# Variable
Python code:
```python
model.add_m_variables(shape[, lb=-inf, ub=+inf, domain=pyoptinterface.VariableDomain.Continuous, name=""])
```
Explanation:
Add a multidimensional variable to the model as numpy.ndarray.
shape – the shape of the variable, can be a tuple of integers or an integer; lb (float) – the lower bound of the variable, optional, defaults to -inf; ub (float) – the upper bound of the variable, optional, defaults to +inf; domain (pyoptinterface.VariableDomain) – the domain of the variable, optional, defaults to continuous; name (str) – the name of the variable, optional.
Returns the multidimensional variable as numpy.ndarray
Example:
```python
Pi = model.add_m_variables((T, bus_num), name="Pi")
Pij = model.add_m_variables((T, branch_num), name="Pij")
u = model.add_m_variables((T, BESS_num), domain=poi.VariableDomain.Binary, name="u")
```

# Linear Constraint
Math format:
\(\text{expr} = a^T x + b \leq \text{rhs} \)
\(\text{expr} = a^T x + b = \text{rhs} \)
\(\text{expr} = a^T x + b \geq \text{rhs} \)
Python code:
```python
model.add_linear_constraint(expr, sense, rhs[, name=""])
```
Explanation:
Add a linear constraint to the model.
expr – the expression of the constraint; sense (pyoptinterface.ConstraintSense) – the sense of the constraint; rhs (float) – the right-hand side of the constraint; name (str) – the name of the constraint, optional.
Returns the handle of the constraint. Note PyOptInterface provides pyoptinterface.Eq, pyoptinterface.Leq, and pyoptinterface.Geq as alias of pyoptinterface.ConstraintSense to represent the sense of the constraint with a shorter name.
Example:
```python
model.add_linear_constraint(Pg[t][i], poi.Leq, P_DG_max, name=f"DG_active_power_max_{t}_{i}")
model.add_linear_constraint(Pg[t][i], poi.Geq, P_DG_min, name=f"DG_active_power_min_{t}_{i}")
model.add_linear_constraint(-SOC[t][BESS] + SOC[t - 1][BESS] - Pg[t][i] / eta + Pc[t][i] * eta, poi.Eq, 0.0, name=f"BESS_SOC_transition_{t}_{i}")
```
(next page)

# Quadratic Constraint
Math format:
\(\text{expr} = x^TQx + a^Tx + b \leq \text{rhs} \)
\(\text{expr} = x^TQx + a^Tx + b = \text{rhs} \)
\(\text{expr} = x^TQx + a^Tx + b \geq \text{rhs} \)
Python code:
```python
model.add_quadratic_constraint(expr, sense, rhs[, name=""])
```
Explanation:
Add a quadratic constraint to the model.
expr – the expression of the constraint; sense (pyoptinterface.ConstraintSense) – the sense of the constraint, which can be GreaterEqual, Equal, or LessEqual; rhs (float) – the right-hand side of the constraint; name (str) – the name of the constraint, optional.
Returns the handle of the constraint.
Example:
```python
model.add_quadratic_constraint(Pg[t][i] * Pg[t][i] + Qg[t][i] * Qg[t][i], poi.Leq, S_PV_max ** 2, name=f"PV_capacity_max_{t}_{i}")
model.add_quadratic_constraint(Pij[t][ij] * Pij[t][ij] + Qij[t][ij] * Qij[t][ij], poi.Leq, S_branch_max ** 2, name=f"branch_power_max_{t}_{ij}")
```

# Second Order Cone Constraint
Math format:
\(variables=(t,x) \in \mathbb{R}^{N} : t \ge \lVert x \rVert_2 \)
Python code:
```python
model.add_second_order_cone_constraint(variables[, name="", rotated=False])
```
Explanation:
Add a second order cone constraint to the model.
variables – the variables of the constraint, can be a list of variables; name (str) – the name of the constraint, optional; rotated (bool) – whether the constraint is a rotated second-order cone constraint, optional.
Returns the handle of the constraint.
Example:
```python
model.add_second_order_cone_constraint([l_plus_v[t][ij], Pij2[t][ij], Qij2[t][ij], l_minus_v[t][ij]], name=f"SOCP_{t}_{ij}")
```

# Objective
```python
model.set_objective(expr[, sense=pyoptinterface.ObjectiveSense.Minimize])
```
Explanation:
Set the objective function of the model.
expr – the handle of the expression; sense (pyoptinterface.ObjectiveSense) – the sense of the objective function (Minimize/Maximize), defaults to Minimize.
Example:
```python
model.set_objective(expr, poi.ObjectiveSense.Minimize)
```
(next page)

~~~
# Optimization
```python
model.optimize()
```
Explanation:
Use the optimizer to optimize the formulated constrained optimization problem

# Get Results
```python
model.get_value(expr_or_var)
```
Explanation:
Get the value of an expression or a variable after optimization.
expr_or_var – the handle of the expression or the variable.
Returns the value of the expression or the variable.
Example:
```python
model.get_value(Pi[t][i])
model.get_value(Pij[t][ij])
model.get_value(u[t][BESS])
```
~~~

Few-shot Examples

~~~
Here is one of the three examples:

The constrained optimization problem of this example is:
<Optimization_Problem>

The Python code of this example is:
```python
<Python_Code>
```
~~~

Complete Code

~~~
The user request is:
<User_Request>

Here is the constrained optimization problem in math format:
<Optimization_Problem_Math_Format>

According to the above information, convert the problem into Python code with PyOptInterface. Your output format is:
<Python_Code>
```python
import pyoptinterface as poi
from pyoptinterface import gurobi
import numpy as np
from casexx import pf_case as mycase (next page)
~~~

```
class Auto_Dispatch:
        def __init__(self, case):
                self.case = case

                # load parameter
                self.T = self.case["T"]
                self.current_t = self.case["current_t"]
                self.BESS_num = self.case["BESS_num"]
                self.MVA = self.case["MVA"]
                self.rho_DG = self.case["rho_DG"]
                self.rho_BESS_dis = self.case["rho_BESS_dis"]
                self.rho_BESS_cha = self.case["rho_BESS_cha"]
                self.rho_0 = self.case["rho_0"]
                self.bus_num = self.case["bus"].shape[0]
                self.branch_num = self.case["branch"].shape[0]
                self.gen_num = self.case["gen"].shape[0]
                self.P_DG_max = self.case["P_DG_max"]
                self.P_DG_min = self.case["P_DG_min"]
                self.Q_DG_max = self.case["Q_DG_max"]
                self.Q_DG_min = self.case["Q_DG_min"]
                self.R_max = self.case["R_max"]
                self.S_PV_max = self.case["S_PV_max"]
                self.P_BESS_dis_max = self.case["P_BESS_dis_max"]
                self.P_BESS_cha_max = self.case["P_BESS_cha_max"]
                self.SOC_BESS_max = self.case["SOC_BESS_max"]
                self.SOC_BESS_min = self.case["SOC_BESS_min"]
                self.SOC_init = self.case["SOC_init"]
                self.Q_SVC_max = self.case["Q_SVC_max"]
                self.eta = self.case["eta"]
                self.V_max = self.case["V_max"]
                self.V_min = self.case["V_min"]
                self.S_branch_max = self.case["S_branch_max"]

                # load model
                self.fbus = self.case["branch"][:, 0].astype(int) - 1
                self.tbus = self.case["branch"][:, 1].astype(int) - 1
                self.R = self.case["branch"][:, 2]
                self.X = self.case["branch"][:, 3]
                self.DG_bus_index = self.case["DG_bus_index"]
                self.PV_bus_index = self.case["PV_bus_index"]
                self.BESS_bus_index = self.case["BESS_bus_index"]
                self.SVC_bus_index = self.case["SVC_bus_index"]

                # load data
                self.PV = self.case["PV"]
                self.P_load = self.case["P_load"]
                self.Q_load = self.case["Q_load"]

                # create optimization model
                self.model = gurobi.Model() (next page)
```

~~~
    def create_variables(self):
        # common variables
        self.Pi = self.model.add_m_variables((self.T, self.bus_num), name="Pi")
        self.Qi = self.model.add_m_variables((self.T, self.bus_num), name="Qi")
        self.Pij = self.model.add_m_variables((self.T, self.branch_num), name="Pij")
        self.Qij = self.model.add_m_variables((self.T, self.branch_num), name="Qij")
        self.Pg = self.model.add_m_variables((self.T, self.gen_num), name="Pg")
        self.Pc = self.model.add_m_variables((self.T, self.gen_num), name="Pc")
        self.Qg = self.model.add_m_variables((self.T, self.gen_num), name="Qg")
        self.v = self.model.add_m_variables((self.T, self.bus_num), name="v")
        self.l = self.model.add_m_variables((self.T, self.branch_num), name="l")
        self.SOC = self.model.add_m_variables((self.T, self.BESS_num), name="SOC")
        self.u = self.model.add_m_variables((self.T, self.BESS_num), domain=poi.VariableDomain.Binary,
name="u")

        # variables for SOCP relaxation
        self.l_plus_v = self.model.add_m_variables((self.T, self.branch_num), name="l_plus_v", lb=0.0)
        self.Pij2 = self.model.add_m_variables((self.T, self.branch_num), name="Pij2")
        self.Qij2 = self.model.add_m_variables((self.T, self.branch_num), name="Qij2")
        self.l_minus_v = self.model.add_m_variables((self.T, self.branch_num), name="l_minus_v")

    def create_constraints(self):
        # The constraints you created

    def create_objective(self):
        # The objective you created


if __name__ == '__main__':
    opt = Auto_Dispatch(mycase())
    opt.create_variables()
    opt.create_constraints()
    opt.create_objective()
    opt.model.optimize()
    Pij = np.zeros((opt.T, opt.branch_num))
    Qij = np.zeros((opt.T, opt.branch_num))
    Pg = np.zeros((opt.T, opt.gen_num))
    Pc = np.zeros((opt.T, opt.gen_num))
    Qg = np.zeros((opt.T, opt.gen_num))
    v = np.zeros((opt.T, opt.bus_num))
    SOC = np.zeros((opt.T, opt.BESS_num))
    for t in range(opt.T):
        for ij in range(opt.branch_num):
            Pij[t][ij] = opt.model.get_value(opt.Pij[t][ij])
            Qij[t][ij] = opt.model.get_value(opt.Qij[t][ij])
        for i in range(opt.gen_num):
            Pg[t][i] = opt.model.get_value(opt.Pg[t][i])
            Pc[t][i] = opt.model.get_value(opt.Pc[t][i])
            Qg[t][i] = opt.model.get_value(opt.Qg[t][i])
        for i in range(opt.bus_num):
            v[t][i] = np.sqrt(opt.model.get_value(opt.v[t][i]))
        for BESS in range(opt.BESS_num):
            SOC[t][BESS] = opt.model.get_value(opt.SOC[t][BESS])
    obj = opt.model.get_model_attribute(poi.ModelAttribute.ObjectiveValue)
```
</Python_Code> (next page)
~~~

Remember:
1. When the district in user request is Valley District, you should "from case_valley import pf_case as mycase"; When the district in user request is Railway District, you should "from case_railway import pf_case as mycase"; When the district in user request is Business District, you should "from case_business import pf_case as mycase".
2. You should fill the "create_constraints" function and "create_objective" function with the given information. You must follow the given examples, do not write the codes on your own!
3. Except for the "case" and "create_constraints" "create_objective" functions, no modifications are needed for other parts of the output format, you must not change anything!
4. You must follow the given output format, the final code should start with
<Python_Code>
```python

and end with
```
</Python_Code>

Also, you can only change the "case" and "create_constraints" "create_objective" functions, nothing else should be modified!

V. Dispatch Requests

Valley District

Hey, can you try to cut costs in Valley District tomorrow? Use the diesel generators, solar, and batteries if you can. Just make sure nothing blows up voltage-wise or overheats the lines.

We're trying to lose less power in Valley District tomorrow. How about scheduling those solar panels and SVCs? Don't let the voltage go beyond limits, yeah?

For Valley District, we need to smooth out voltage swings tomorrow. Use DGs, solar, and SVC, don't forget the branch power flow limitation.

Valley District needs a budget-friendly plan. DGs, PVs, and SVCs only. And don't let the voltage cross the upper and lower limit. The branch power should also be safe enough.

Let's save cash in Valley District tomorrow. DGs, batteries, solar. Just don't blow the voltage or overheat the lines.

Valley District's PV is dead tomorrow. Use all other devices to minimize voltage fluctuations.

Voltage in Valley District just spiked above 1.05—can you fix this ASAP? Use SVCs and PVs, but don't let the lines catch fire.

Valley District's branch is overloaded and climbing. PVs, and batteries to the rescue. Remember the voltage at the same time.

Valley District's voltage drops. PVs and SVCs inject reactive power. Now.

Valley District's voltage at 0.89 p.u.. PV and SVCs boost, batteries discharge. Hurry.

Railway District

Please minimize the operational cost of Railway District for tomorrow. Resources: photovoltaics, diesel generators, battery storage. Note: Maintain voltage safety and prevent branch overloads.

Tomorrow, we need to minimize power loss in Railway District. Schedule PV systems and SVCs while ensuring voltage remains within ±5% of nominal.

For Railway District, reduce operational costs for tomorrow. All assets (DGs, BESS, PV, SVC) are available. Remember the voltage safety and branch power safety.

Plan for minimal network loss in Railway District. Prioritize PV and SVC deployment. Avoid branch power violations.

For Railway District, minimize voltage fluctuations across all feeders. Use all available resources except BESS.

Tomorrow, Railway District requires cost-efficient dispatch. PVs and BESS only. Prevent overvoltage and lower voltage, also prevent branch over flow.

Railway District: Voltage exceeds 1.05 p.u. Urgently activate PV absorb reactive power and BESS to absorb active power!

Overload detected in Railway District branch. Use DGs, PVs, and BESS to reroute power. Prioritize voltage safety.

Railway District reports under-voltage (0.92 p.u.). Deploy BESS discharge and SVC inductive compensation immediately.

Railway District: Immediate action needed for 120% branch overload. Mobilize DGs, PVs and BESS.

Business District

Please minimize the operational cost for Business District tomorrow by optimally dispatching diesel generators, battery storage systems, and photovoltaic units. Ensure voltage levels remain within ±5% of nominal values and branch loading does not exceed rated capacity.

We kindly request a plan to reduce network losses in Business District for the upcoming operational day. Utilize photovoltaic systems and static VAR compensators while maintaining voltage safety and avoiding branch thermal limits.

Could you prioritize voltage deviation minimization in Business District tomorrow? Coordinate diesel generators, photovoltaics, and SVCs.

Please optimize resource scheduling for Business District to minimize operational costs. Leverage all of the equipment, ensuring no power flows violation and voltage remains within acceptable bounds.

We require a strategy to reduce power losses in Business District for tomorrow. Focus on photovoltaic and SVC deployment, adhering to voltage constraints.

Could you design a day-ahead plan for Business District to minimize voltage deviations? Integrate photovoltaic systems, SVCs, and DGs while avoiding branch overloads.

Immediate action is required: Voltage in Business District has exceeded 0.95 p.u. Please utilize static VAR compensators and photovoltaic inverters to restore nominal voltage levels.

An overload condition has been detected in Business District. Please dispatch diesel generators, photovoltaics, and battery storage systems to alleviate branch congestion while ensuring voltage safety.

Business District is experiencing undervoltage (0.92 p.u.). Urgently deploy battery storage discharge and SVC capacitive support to stabilize voltages.

Business District reports a 120% branch overload. Please mobilize diesel generators, PVs, and battery discharge to eliminate thermal violations. Remember voltage safety.

References

[1] X. Yang, C. Lin, H. Liu and W. Wu, "RL2: Reinforce Large Language Model to Assist Safe Reinforcement Learning for Energy Management of Active Distribution Networks," *IEEE Trans. Smart Grid*, early access.

[2] M. Jia, Z. Cui, and G. Hug, "Enhancing LLMs for Power System Simulations: A Feedback-driven Multi-agent Framework," *arXiv:2411.16707*, May 2025, [Online]. Available: https://arxiv.org/abs/2411.16707.

[3] Y. Yang, C. Lin, L. Xu et al., "PyOptInterface: Design and implementation of an efficient modeling language for mathematical optimization," *arXiv: 2405.10130*, May 2024, [Online]. Available: https://arxiv.org/abs/2405.10130.