**4. Run Conventional Grid Analysis**

This step performs a baseline “industry‐standard” assessment of your blended topology: first by verifying the system state under normal conditions, then by screening contingencies and prescribing remedial measures.

**4.1 State Estimation (WLS / NR)**

1. **Measurement Model**
   * Collect “pseudo‑measurements” from your mock SCADA/PMU streams:

z=[PijQijViθi]  =  h(x)  +  e z = \begin{bmatrix} P\_{ij} \\ Q\_{ij} \\ V\_i \\ \theta\_i \end{bmatrix} \;=\; h(x) \;+\; ez=​Pij​Qij​Vi​θi​​​=h(x)+e

* + Where x=[θ, V]x=[\theta,\,V]x=[θ,V] is the system state, h(x)h(x)h(x) the nonlinear measurement function, and eee Gaussian noise.

1. **Weighted Least Squares Formulation**
   * Define the objective

J(x)=[ z−h(x) ]TW−1[ z−h(x) ] J(x) = [\,z - h(x)\,]^T W^{-1} [\,z - h(x)\,]J(x)=[z−h(x)]TW−1[z−h(x)]

* + WWW is the diagonal weight matrix (inverse variances of each measurement).

1. **Newton–Raphson Iteration**
   * Initialize x(0)x^{(0)}x(0) (flat start or last good state).
   * Repeat until ∥Δx∥\|\Delta x\|∥Δx∥ < ε or max iters:
     1. Compute residual r=z−h(x(k))r = z - h(x^{(k)})r=z−h(x(k))
     2. Form Jacobian H=∂h∂x∣x(k)H = \frac{\partial h}{\partial x}\big|\_{x^{(k)}}H=∂x∂h​​x(k)​
     3. Solve normal equations:

(HTW−1H) Δx=HTW−1r (H^T W^{-1} H)\,\Delta x = H^T W^{-1} r(HTW−1H)Δx=HTW−1r

* + 1. Update x(k+1)=x(k)+Δxx^{(k+1)} = x^{(k)} + \Delta xx(k+1)=x(k)+Δx

1. **Bad‑Data Detection**
   * Compute normalized residuals ri/(diag(S))ir\_i/\sqrt{(\mathrm{diag}(S))\_i}ri​/(diag(S))i​​, where  
     S=H(HTW−1H)−1HTS = H (H^T W^{-1} H)^{-1} H^TS=H(HTW−1H)−1HT.
   * If any exceed threshold (e.g., 3σ), flag & remove that measurement, then re‑estimate.
2. **Outcome**
   * A “clean” state estimate x^\hat xx^ (voltage magnitudes & angles) under normal operation.
   * Store x^\hat xx^ for reference and as the “base case” for contingency runs.

**4.2 Real‑Time Contingency Analysis (RTCA)**

1. **Scenario Loop**  
   For each contingency scenario (from Section 3):

python

CopyEdit

for cont in contingencies:

net = base\_pp\_net.deepcopy()

cont.apply(net) # remove line/gen/xfmr

runpf(net) # steady‑state PF

violations = detect\_violations(net)

record\_results(cont, violations)

1. **Violation Detection**
   * **Thermal Limits**: any branch’s loading > rating\_normal
   * **Voltage Violations**: Vi∉[0.95,1.05]V\_i\notin[0.95,1.05]Vi​∈/[0.95,1.05] pu
   * **Island Checks**: loss of connectivity or radial splits
2. **Violation Ranking**
   * Rank by a **severity index**, e.g.:

SI=w1⋅(% overload)+w2⋅(voltage deviation)+w3⋅(customers affected) \mathrm{SI} = w\_1\cdot(\%\,\text{overload}) + w\_2\cdot(\text{voltage deviation}) + w\_3\cdot(\text{customers affected})SI=w1​⋅(%overload)+w2​⋅(voltage deviation)+w3​⋅(customers affected)

**4.3 Documenting Remedial Actions**

For each contingency that causes violations, apply and log a **rule‑based** mitigation, e.g.:

| **Violation Type** | **Mitigation** | **Action Detail** |
| --- | --- | --- |
| Line Overload | Generator redispatch | Ramp up gen at nearest bus by ΔMW |
| Voltage Drop | FACTS tuning or tap‑changer adjustment | Increase transformer tap at Substation X |
| Islanded Section | Automatic load shedding | Shed 10% load at highest‑impact feeders |
| Loss of Reserve | Activate spinning reserve | Dispatch BESS or fast‑start gas unit |

1. **Apply** the chosen action to the net object.
2. **Re‑run PF** to confirm the violation is cleared or sufficiently mitigated.
3. **Log** in your results structure:

python

CopyEdit

{

"cont\_id": cont.id,

"violation": (element, type, severity),

"mitigation": (action\_type, target, amount\_mw),

"post\_status": "cleared" or "residual\_violation"

}

**4.4 Output for Excel**

* **Compile** all recorded results into a pandas DataFrame with columns:

csharp

CopyEdit

[cont\_id, element\_id, violation\_type, severity, mitigation\_type,

target\_id, amount\_mw, post\_status]

* **Export** via

python

CopyEdit

df.to\_excel("contingency\_and\_mitigation\_results.xlsx", index=False)

* This file then feeds into your downstream ML experiments.