# ieee_33_bus_system

July 15, 2025

# 1 Neuro-Fuzzy Load Flow Estimation for Disaster-Resilient Smart Grids

This repository contains the implementation of a B.Tech project focused on **neuro-fuzzy load flow estimation** for **disaster-resilient smart grids**, simulating mobile sensor data for the IEEE 33-bus distribution systems.

The project develops a neuro-fuzzy model to estimate grid states (voltage magnitudes and angles) using sparse, noisy measurements from **mobile sensors** (e.g., IoT devices or drones) in **post-disaster scenarios** (e.g., earthquakes), where fixed sensors may fail. The dataset generation script simulates realistic sensor behavior with **5–10% noise** and **30–50% missing data**, enabling robust training of the model.

## 1.1 Contributors

- **Abhinav Jha (2K22/EE/10)** — Led data generation, power system modeling, and hardware design
- **Akshin Saxena (2K22/EE/36)** — Focused on digital logic and data preprocessing
- **Akshat Garg (2K22/EE/35)** — Contributed to signal processing and dataset validation

**Date**: July 2025

## 1.2 Project Overview

The project addresses the challenge of load flow estimation in smart grids under **disaster conditions**, where traditional SCADA systems may be unreliable due to physical damage or communication failures.

Mobile sensors are simulated to collect **sparse measurements** (voltage, current, power flow) at a subset of buses or lines in the IEEE 33-bus systems (12.66 kV). A **neuro-fuzzy model** (combining fuzzy logic and ANNs) processes these measurements to predict full grid states, even under **30–50% data sparsity**.

# 2 Data Loading and Inspection

This cell loads the generated CSV files (`sensor_inputs_ieee33-bus.csv` and `grid_states_ieee33-bus.csv`) and displays their shapes, column names, and the first few rows to verify the dataset structure. The `sensor_inputs_ieee33-bus.csv` contains sparse, noisy measurements (voltages, currents, power flows) from 5–10 sensors per scenario. The `grid_states_ieee33-bus.csv` contains full grid states (33 voltage magnitudes and 33 angles) for 5,000 scenarios.

[ ]:
```
import pandas as pd  # type: ignore
import matplotlib.pyplot as plt # type: ignore
import numpy as np # type: ignore
import seaborn as sns # type: ignore

# Load CSV Files
inputs = pd.read_csv('output_generation/sensor_inputs_ieee_33-bus.csv');
states = pd.read_csv('output_generation/grid_states_ieee_33-bus.csv');

#Display shapes
print(f"Sensor Inputs Shape: {inputs.shape}")
print(f"Grid States Shape: {inputs.shape}")
```

```
Sensor Inputs Shape: (5000, 20)
Grid States Shape: (5000, 20)
```

[2]:
```
# Display column names\n",
print("Sensor Inputs Column: ", list(inputs.columns))
print("Grid States Column: ", list(states.columns))
```

```
Sensor Inputs Column:  ['meas_0', 'meas_1', 'meas_2', 'meas_3', 'meas_4',
'meas_5', 'meas_6', 'meas_7', 'meas_8', 'meas_9', 'meas_10', 'meas_11',
'meas_12', 'meas_13', 'meas_14', 'meas_15', 'meas_16', 'meas_17', 'meas_18',
'meas_19']
Grid States Column:  ['V_0', 'V_1', 'V_2', 'V_3', 'V_4', 'V_5', 'V_6', 'V_7',
'V_8', 'V_9', 'V_10', 'V_11', 'V_12', 'V_13', 'V_14', 'V_15', 'V_16', 'V_17',
'V_18', 'V_19', 'V_20', 'V_21', 'V_22', 'V_23', 'V_24', 'V_25', 'V_26', 'V_27',
'V_28', 'V_29', 'V_30', 'V_31', 'V_32', 'theta_0', 'theta_1', 'theta_2',
'theta_3', 'theta_4', 'theta_5', 'theta_6', 'theta_7', 'theta_8', 'theta_9',
'theta_10', 'theta_11', 'theta_12', 'theta_13', 'theta_14', 'theta_15',
'theta_16', 'theta_17', 'theta_18', 'theta_19', 'theta_20', 'theta_21',
'theta_22', 'theta_23', 'theta_24', 'theta_25', 'theta_26', 'theta_27',
'theta_28', 'theta_29', 'theta_30', 'theta_31', 'theta_32']
```

[3]:
```
# Display first 5 rows\n",
print("Sensor Inputs Sample:");
print(inputs.head());
print("Grid States Sample:");
print(states.head())
```

```
Sensor Inputs Sample:
     meas_0    meas_1  meas_2     meas_3     meas_4    meas_5    meas_6  \
0  0.969247  0.995589     NaN  15.682318        NaN  0.118461       NaN
1  0.948872  0.999119     NaN   0.947498  12.196964  0.231715       NaN
```

```
2       NaN  0.987790     NaN        NaN   1.887449   1.395897       NaN
3  0.934559       NaN     NaN   0.925646        NaN   0.495930       NaN
4       NaN  0.904419     NaN   0.742981   0.365112  14.901183  0.241003

     meas_7      meas_8      meas_9     meas_10   meas_11   meas_12  \
0       NaN         NaN  127.872793    2.144022  1.544388       NaN
1       NaN         NaN         NaN         NaN       NaN       NaN
2       NaN    0.038912   51.986091         NaN  0.662265  2.061627
3       NaN    0.533832    0.254429  187.323524  3.636548       NaN
4  0.089147  109.792136    1.921337    1.143176       NaN       NaN

     meas_13   meas_14   meas_15    meas_16   meas_17  meas_18  meas_19
0        NaN       NaN       NaN        NaN       NaN      NaN      NaN
1  24.788501  0.545216  0.225464        NaN       NaN      NaN      NaN
2        NaN       NaN       NaN        NaN       NaN      NaN      NaN
3        NaN       NaN  0.322046  49.893632  0.999282      NaN      NaN
4        NaN       NaN       NaN        NaN       NaN      NaN      NaN
Grid States Sample:
   V_0       V_1       V_2       V_3       V_4       V_5       V_6       V_7  \
0  1.0  0.997281  0.984418  0.977440  0.970659  0.953716  0.950656  0.938275
1  1.0  0.997647  0.986698  0.980575  0.974540  0.959878  0.956935  0.945367
2  1.0  0.997428  0.985702  0.979594  0.973570  0.959215  0.956086  0.943379
3  1.0  0.997299  0.985449  0.979453  0.973493  0.959452  0.956019  0.940888
4  1.0  0.997327  0.985713  0.979277  0.972990  0.957854  0.953796  0.935973

        V_8       V_9  …  theta_23  theta_24  theta_25  theta_26  theta_27  \
0  0.931875  0.926120  …  0.012783 -0.024795  0.258125  0.318985  0.431624
1  0.939842  0.935050  …  0.005836 -0.018885  0.141368  0.185596  0.264285
2  0.937378  0.932243  … -0.013284 -0.050109  0.152304  0.196495  0.282217
3  0.934173  0.928677  … -0.043150 -0.091512  0.032387  0.063271  0.113254
4  0.928828  0.922355  … -0.048808 -0.086275 -0.075615 -0.057760 -0.061889

   theta_28  theta_29  theta_30  theta_31  theta_32
0  0.529033  0.642605  0.571501  0.547994  0.542875
1  0.335627  0.418401  0.367849  0.350589  0.345333
2  0.356070  0.438191  0.390835  0.374205  0.367149
3  0.161003  0.218811  0.183552  0.171215  0.165092
4 -0.056719 -0.023155 -0.073772 -0.091268 -0.098632

[5 rows x 66 columns]
```

# 3 Sparsity Analysis

This cell calculates the percentage of missing (`NaN`) values in `sensor_inputs_ieee33-bus.csv` to verify the 30–50% sparsity requirement, simulating disaster conditions where sensor data may be unavailable. The overall sparsity and per-column sparsity are computed to show the dataset's robustness for neuro-fuzzy modeling.

[4]:
```python
# Calculate overall sparsity
overall_sparsity = inputs.isna().mean().mean() * 100
print(f"Overall Sparsity: {overall_sparsity:.2f}%")
```

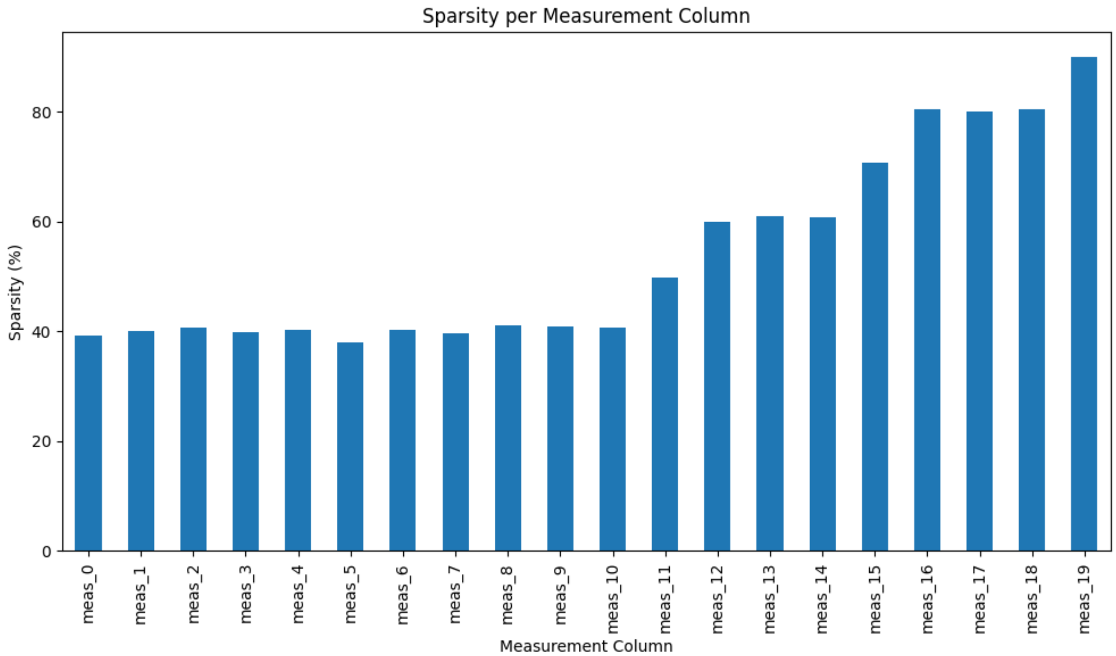
```
Overall Sparsity: 53.67%
```

[5]:
```python
# Calculate per-column sparsity
column_sparsity = inputs.isna().mean() * 100
print("\nPer-Column Sparsity (%):")
print(column_sparsity)
```

```

Per-Column Sparsity (%):
meas_0     39.18
meas_1     40.06
meas_2     40.72
meas_3     39.76
meas_4     40.16
meas_5     38.00
meas_6     40.30
meas_7     39.58
meas_8     41.10
meas_9     40.94
meas_10    40.58
meas_11    49.80
meas_12    60.00
meas_13    60.96
meas_14    60.68
meas_15    70.66
meas_16    80.46
meas_17    79.98
meas_18    80.46
meas_19    89.98
dtype: float64
```

[6]:
```python
# Plot sparsity per column
plt.figure(figsize=(10, 6))
column_sparsity.plot(kind='bar')
plt.title('Sparsity per Measurement Column')
plt.xlabel('Measurement Column')
plt.ylabel('Sparsity (%)')
plt.tight_layout()
plt.savefig('sparsity_plot.png')
plt.show()
```




# 4 statistical_summaries.py

Computes statistical summaries for sensor inputs and grid states.

[7]:
```
# Summary for sensor inputs
print("Sensor Inputs Summary:")
print(inputs.describe())
```

```
Sensor Inputs Summary:
            meas_0       meas_1       meas_2        meas_3        meas_4  \
count  3041.000000  2997.000000  2964.000000  3.012000e+03  2.992000e+03
mean      0.999696     0.998281     0.884285  7.312915e-01  2.704552e-01
std       0.077122     0.077491     1.005096  3.718524e+00  2.543000e+00
min       0.659934     0.619800     0.000000 -6.543725e-30 -3.881445e-10
25%       0.950660     0.948387     0.908805  0.000000e+00  0.000000e+00
50%       0.998035     0.997456     0.983179  8.231513e-01  0.000000e+00
75%       1.049067     1.048706     1.039241  1.002155e+00  6.196533e-07
max       1.259750     1.282208    36.745505  1.372228e+02  1.016789e+02

             meas_5        meas_6        meas_7        meas_8        meas_9  \
count  3.100000e+03  2.985000e+03  3.021000e+03  2.945000e+03  2.953000e+03
mean   1.418263e-01  2.098799e-01  1.173625e-01  1.528000e-01  1.614159e-01
std    2.468302e+00  4.420161e+00  2.319739e+00  3.206974e+00  2.985413e+00
min   -2.970444e-09 -3.453138e-09 -1.255138e-09 -1.216271e-09 -4.504534e-09
25%    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
50%    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
```

```
75%    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
max    7.902508e+01  2.185002e+02  9.862012e+01  1.127597e+02  1.278728e+02

            meas_10       meas_11       meas_12       meas_13       meas_14  \
count  2.971000e+03  2.510000e+03  2.000000e+03  1.952000e+03  1.966000e+03
mean   1.935711e-01  8.273384e-02  1.946629e-01  3.494617e-01  1.932563e-01
std    4.144890e+00  2.179756e+00  3.797951e+00  5.421486e+00  3.618023e+00
min   -2.157594e-09 -1.497360e-09 -1.355155e-09 -7.601166e-10 -1.178505e-09
25%    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
50%    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
75%    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
max    1.873235e+02  1.010039e+02  1.140553e+02  1.585008e+02  1.311534e+02

            meas_15       meas_16       meas_17       meas_18       meas_19
count  1.467000e+03  9.770000e+02  1.001000e+03  9.770000e+02  5.010000e+02
mean   9.995063e-03  2.086595e-01  2.434531e-01  4.605047e-03  4.767422e-03
std    8.591807e-02  2.568516e+00  3.275283e+00  6.760398e-02  6.141308e-02
min   -1.558185e-09 -2.501155e-09 -8.483685e-27 -8.640797e-10 -1.554651e-09
25%    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
50%    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
75%    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
max    1.645317e+00  4.989363e+01  8.352939e+01  1.911378e+00  1.262258e+00
```

[8]:

```python
# Summary for voltage magnitudes (V_0 to V_32)
voltage_cols = [col for col in states.columns if col.startswith('V_')]
print("\nVoltage Magnitudes Summary (pu):")
print(states[voltage_cols].describe())
```

```
Voltage Magnitudes Summary (pu):
          V_0          V_1          V_2          V_3          V_4  \
count  5000.0  5000.000000  5000.000000  5000.000000  5000.000000
mean      1.0     0.999960     0.999854     0.999798     0.999744
std       0.0     0.000270     0.001102     0.001556     0.002004
min       1.0     0.995255     0.984418     0.977440     0.970659
25%       1.0     1.000000     1.000000     1.000000     1.000000
50%       1.0     1.000000     1.000000     1.000000     1.000000
75%       1.0     1.000000     1.000000     1.000000     1.000000
max       1.0     1.000000     1.000000     1.000000     1.000000

               V_5          V_6          V_7          V_8          V_9  …  \
count  5000.000000  5000.000000  5000.000000  5000.000000  5000.000000  …
mean      0.999611     0.999578     0.999427     0.999348     0.999270  …
std       0.003110     0.003370     0.004551     0.005167     0.005775  …
min       0.953716     0.950656     0.935973     0.928828     0.922355  …
25%       1.000000     1.000000     1.000000     1.000000     1.000000  …
50%       1.000000     1.000000     1.000000     1.000000     1.000000  …
75%       1.000000     1.000000     1.000000     1.000000     1.000000  …
```

```
max       1.000000     1.000000     1.000000     1.000000     1.000000  …

              V_23         V_24         V_25         V_26         V_27  \
count  5000.000000  5000.000000  5000.000000  5000.000000  5000.000000
mean      0.999778     0.999762     0.999600     0.999586     0.999522
std       0.001808     0.001953     0.003205     0.003333     0.003897
min       0.974388     0.972576     0.951847     0.949362     0.938331
25%       1.000000     1.000000     1.000000     1.000000     1.000000
50%       1.000000     1.000000     1.000000     1.000000     1.000000
75%       1.000000     1.000000     1.000000     1.000000     1.000000
max       1.000000     1.000000     1.000000     1.000000     1.000000

              V_28         V_29         V_30         V_31         V_32
count  5000.000000  5000.000000  5000.000000  5000.000000  5000.000000
mean      0.999476     0.999453     0.999413     0.999399     0.999386
std       0.004311     0.004509     0.004851     0.004963     0.005074
min       0.930299     0.926885     0.923402     0.922492     0.922303
25%       1.000000     1.000000     1.000000     1.000000     1.000000
50%       1.000000     1.000000     1.000000     1.000000     1.000000
75%       1.000000     1.000000     1.000000     1.000000     1.000000
max       1.000000     1.000000     1.000000     1.000000     1.000000

[8 rows x 33 columns]
```

[9]:
```python
# Summary for voltage angles (theta_0 to theta_32)
angle_cols = [col for col in states.columns if col.startswith('theta_')]
print("\nVoltage Angles Summary (degrees):")
print(states[angle_cols].describe())
```

```
Voltage Angles Summary (degrees):
       theta_0       theta_1       theta_2       theta_3       theta_4  \
count   5000.0  5.000000e+03  5.000000e+03  5.000000e+03  5.000000e+03
mean       0.0  9.788883e-05  4.353435e-04  7.193093e-04  9.825376e-04
std        0.0  8.844776e-04  3.925137e-03  6.666745e-03  9.393909e-03
min        0.0 -5.792180e-04 -8.695918e-04 -1.137028e-03 -1.469727e-03
25%        0.0 -9.196782e-20 -1.678116e-18 -1.913117e-18 -1.953383e-18
50%        0.0 -1.570108e-35 -2.709403e-35 -2.984291e-35 -3.058553e-35
75%        0.0 -1.052155e-58 -3.384347e-58 -3.890465e-58 -4.304860e-58
max        0.0  1.873879e-02  1.112112e-01  1.863968e-01  2.623811e-01

            theta_5       theta_6       theta_7       theta_8       theta_9  \
count  5.000000e+03  5.000000e+03  5.000000e+03  5.000000e+03  5.000000e+03
mean  -2.441540e-04 -2.282508e-03 -3.707694e-03 -4.457241e-03 -5.157013e-03
std    6.336215e-03  1.906103e-02  3.024541e-02  3.623184e-02  4.181343e-02
min   -9.279385e-02 -3.436251e-01 -5.201545e-01 -5.922450e-01 -6.491598e-01
25%   -1.539889e-16 -3.526079e-16 -4.486449e-16 -4.488813e-16 -4.653401e-16
50%   -9.941694e-35 -8.579448e-34 -1.231226e-33 -1.231226e-33 -1.277622e-33
```

```
75%   -5.541168e-57 -2.942018e-55 -4.111108e-55 -4.111108e-55 -4.304514e-55
max    2.152523e-01  1.928426e-02  1.550327e-02  1.487660e-02  1.426418e-02

       …      theta_23      theta_24      theta_25      theta_26  \
count  …  5.000000e+03  5.000000e+03  5.000000e+03  5.000000e+03
mean   … -4.574971e-04 -6.662130e-04 -9.543976e-05  1.154817e-04
std    …  5.529676e-03  7.220537e-03  6.958934e-03  8.303208e-03
min    … -1.183611e-01 -1.372041e-01 -9.036222e-02 -8.690476e-02
25%    … -1.391594e-16 -1.409438e-16 -1.508262e-16 -1.300976e-16
50%    … -1.010392e-34 -1.013541e-34 -9.539166e-35 -8.181429e-35
75%    … -5.773939e-57 -5.991791e-57 -5.182703e-57 -4.073597e-57
max    …  1.866279e-02  1.866273e-02  2.581252e-01  3.189853e-01

           theta_27      theta_28      theta_29      theta_30      theta_31  \
count  5.000000e+03  5.000000e+03  5.000000e+03  5.000000e+03  5.000000e+03
mean   1.308849e-04  2.009242e-04  5.921120e-04  1.504556e-05 -2.464073e-04
std    1.128131e-02  1.395726e-02  1.719439e-02  1.546377e-02  1.528830e-02
min   -8.859941e-02 -1.122247e-01 -9.015892e-02 -1.621083e-01 -1.975297e-01
25%   -1.535170e-16 -1.616784e-16 -1.464512e-16 -1.691995e-16 -1.783710e-16
50%   -9.626633e-35 -1.024419e-34 -9.253957e-35 -1.122406e-34 -1.238044e-34
75%   -5.276427e-57 -6.152818e-57 -5.053953e-57 -6.768154e-57 -7.806165e-57
max    4.316238e-01  5.290328e-01  6.426049e-01  5.715009e-01  5.479937e-01

           theta_32
count  5.000000e+03
mean  -5.745034e-04
std    1.607057e-02
min   -2.606563e-01
25%   -1.831445e-16
50%   -1.286176e-34
75%   -8.196185e-57
max    5.428752e-01

[8 rows x 33 columns]
```

## 5 noise_analysis.py

Analyzes noise in voltage measurements, expecting 5–10% relative error.

[10]:
```python
# Identify voltage measurement columns (mean ~0.95-1.05 pu)
voltage_cols = [col for col in inputs.columns if 0.8 < inputs[col].
    ↪mean(skipna=True) < 1.2]
voltage_cols
```

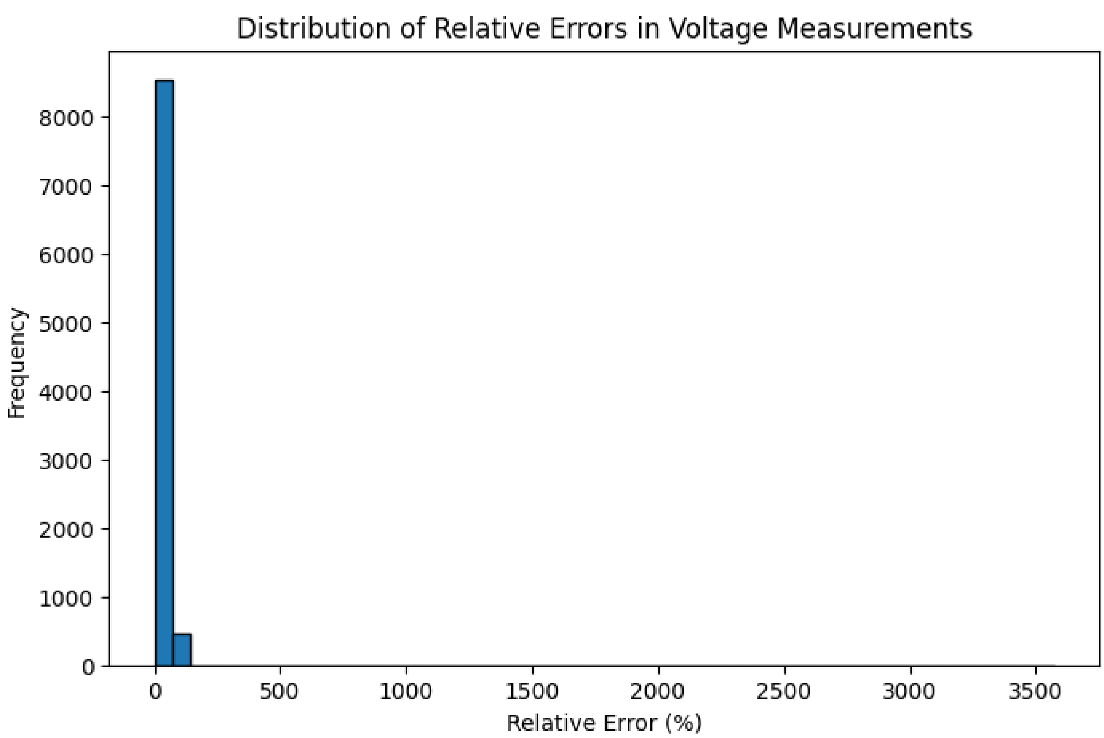
[10]:
```
['meas_0', 'meas_1', 'meas_2']
```

[11]:
```python
# Calculate relative error assuming true voltage ~1.0 pu
relative_errors = []
for col in voltage_cols:
    non_missing = inputs[col].dropna()
    errors = abs(non_missing - 1.0) / 1.0 * 100  # % error
    relative_errors.extend(errors)
```

[12]:
```python
# Print mean and std of relative errors
print(f"Mean Relative Error (Voltage, %): {np.mean(relative_errors):.2f}")
print(f"Std Relative Error (Voltage, %): {np.std(relative_errors):.2f}")
```

```
Mean Relative Error (Voltage, %): 12.03
Std Relative Error (Voltage, %): 57.14
```

[13]:
```python
# Plot histogram of relative errors
plt.figure(figsize=(8, 5))
plt.hist(relative_errors, bins=50, edgecolor='black')
plt.title('Distribution of Relative Errors in Voltage Measurements')
plt.xlabel('Relative Error (%)')
plt.ylabel('Frequency')
plt.savefig('noise_histogram.png')
plt.show()
```

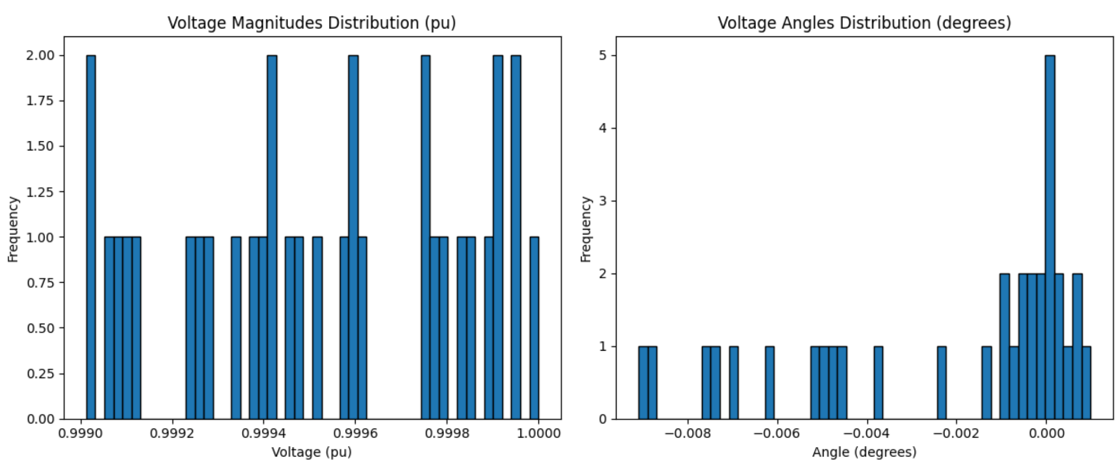


## 6 visualizations.py

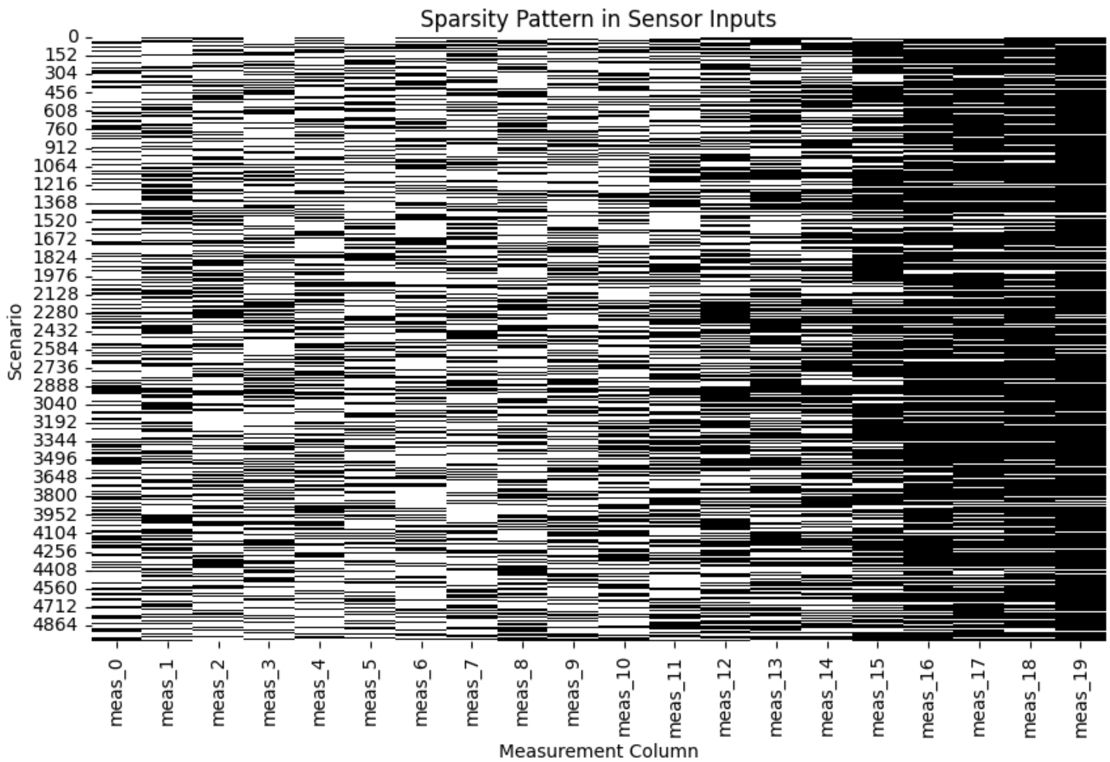
Visualizes dataset characteristics: voltage/angle distributions, sparsity, and voltage profiles.

[14]:
```
# Histogram of voltage magnitudes and angles
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
states[[col for col in states.columns if col.startswith('V_')]].mean().
  ↪plot(kind='hist', bins=50, edgecolor='black')
plt.title('Voltage Magnitudes Distribution (pu)')
plt.xlabel('Voltage (pu)')
plt.ylabel('Frequency')

plt.subplot(1, 2, 2)
states[[col for col in states.columns if col.startswith('theta_')]].mean().
  ↪plot(kind='hist', bins=50, edgecolor='black')
plt.title('Voltage Angles Distribution (degrees)')
plt.xlabel('Angle (degrees)')
plt.ylabel('Frequency')
plt.tight_layout()
plt.savefig('voltage_angle_histograms.png')
plt.show()
```

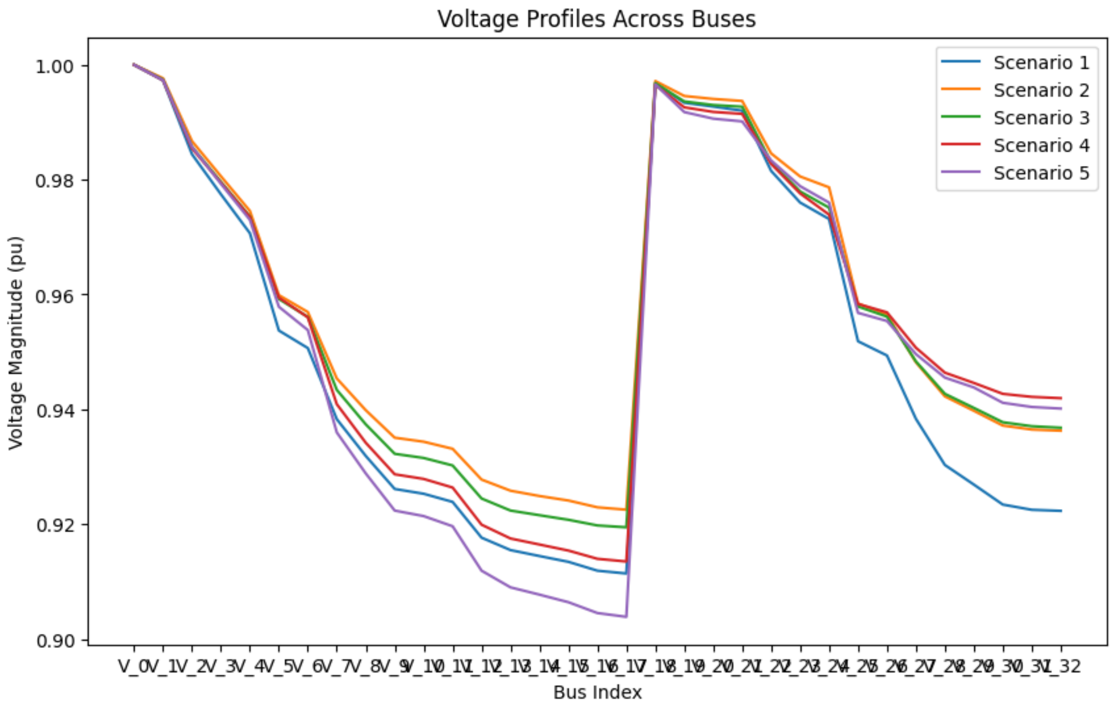



[15]:
```
# Heatmap of missing data
plt.figure(figsize=(10, 6))
sns.heatmap(inputs.isna(), cbar=False, cmap='binary')
plt.title('Sparsity Pattern in Sensor Inputs')
plt.xlabel('Measurement Column')
plt.ylabel('Scenario')
plt.savefig('sparsity_heatmap.png')
plt.show()
```




[16]:

```
# Voltage profiles for 5 scenarios
plt.figure(figsize=(10, 6))
for i in range(5):
    plt.plot(states.iloc[i, :33], label=f'Scenario {i+1}')
plt.title('Voltage Profiles Across Buses')
plt.xlabel('Bus Index')
plt.ylabel('Voltage Magnitude (pu)')
plt.legend()
plt.savefig('voltage_profiles.png')
plt.show()
```




# 7 correlation_analysis.py

Analyzes correlations between sensor inputs and grid states.

[17]:
```python
# Select subset of states (first 5 voltages and angles)
state_subset = states[['V_0', 'V_1', 'V_2', 'V_3', 'V_4', 'theta_0', 'theta_1',␣
↪'theta_2', 'theta_3', 'theta_4']]
state_subset
```

[17]:

| | V_0 | V_1 | V_2 | V_3 | V_4 | theta_0 | theta_1 | \ |
|---|---|---|---|---|---|---|---|---|
| 0 | 1.0 | 0.997281 | 0.984418 | 0.977440 | 0.970659 | 0.0 | 1.683610e-02 | |
| 1 | 1.0 | 0.997647 | 0.986698 | 0.980575 | 0.974540 | 0.0 | 1.168180e-02 | |
| 2 | 1.0 | 0.997428 | 0.985702 | 0.979594 | 0.973570 | 0.0 | 1.152840e-02 | |
| 3 | 1.0 | 0.997299 | 0.985449 | 0.979453 | 0.973493 | 0.0 | 6.856707e-03 | |
| 4 | 1.0 | 0.997327 | 0.985713 | 0.979277 | 0.972990 | 0.0 | 4.398983e-03 | |
| … | … | … | … | … | … | … | … | |
| 4995 | 1.0 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.0 | -6.494008e-76 | |
| 4996 | 1.0 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.0 | -7.126380e-76 | |
| 4997 | 1.0 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.0 | -3.748252e-76 | |
| 4998 | 1.0 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.0 | -3.778475e-76 | |
| 4999 | 1.0 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.0 | -2.379319e-76 | |

| | theta_2 | theta_3 | theta_4 |
|---|---|---|---|
| 0 | 1.112112e-01 | 1.863968e-01 | 2.623811e-01 |

```
1     7.749652e-02  1.302527e-01  1.834786e-01
2     7.753990e-02  1.315715e-01  1.861672e-01
3     4.846301e-02  8.455337e-02  1.205869e-01
4     3.110892e-02  5.452336e-02  7.629549e-02
…              …             …             …
4995 -6.801926e-76 -7.030596e-76 -7.268711e-76
4996 -7.307662e-76 -7.442287e-76 -7.582474e-76
4997 -3.883429e-76 -3.983816e-76 -4.088349e-76
4998 -3.956821e-76 -4.089266e-76 -4.227182e-76
4999 -2.489583e-76 -2.571468e-76 -2.656736e-76

[5000 rows x 10 columns]
```

[18]:
```python
# Combine inputs and states, drop NaN rows
combined = pd.concat([inputs, state_subset], axis=1).dropna()
combined
```

[18]:
```
Empty DataFrame
Columns: [meas_0, meas_1, meas_2, meas_3, meas_4, meas_5, meas_6, meas_7,
meas_8, meas_9, meas_10, meas_11, meas_12, meas_13, meas_14, meas_15, meas_16,
meas_17, meas_18, meas_19, V_0, V_1, V_2, V_3, V_4, theta_0, theta_1, theta_2,
theta_3, theta_4]
Index: []

[0 rows x 30 columns]
```

[19]:
```python
# Compute correlation matrix
corr_matrix = combined.corr()
corr_matrix
```

[19]:
```
         meas_0  meas_1  meas_2  meas_3  meas_4  meas_5  meas_6  meas_7  \
meas_0      NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
meas_1      NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
meas_2      NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
meas_3      NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
meas_4      NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
meas_5      NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
meas_6      NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
meas_7      NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
meas_8      NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
meas_9      NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
meas_10     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
meas_11     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
meas_12     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
meas_13     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
meas_14     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
meas_15     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
```

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| meas_16 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| meas_17 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| meas_18 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| meas_19 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| V_0 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| V_1 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| V_2 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| V_3 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| V_4 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| theta_0 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| theta_1 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| theta_2 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| theta_3 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| theta_4 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |

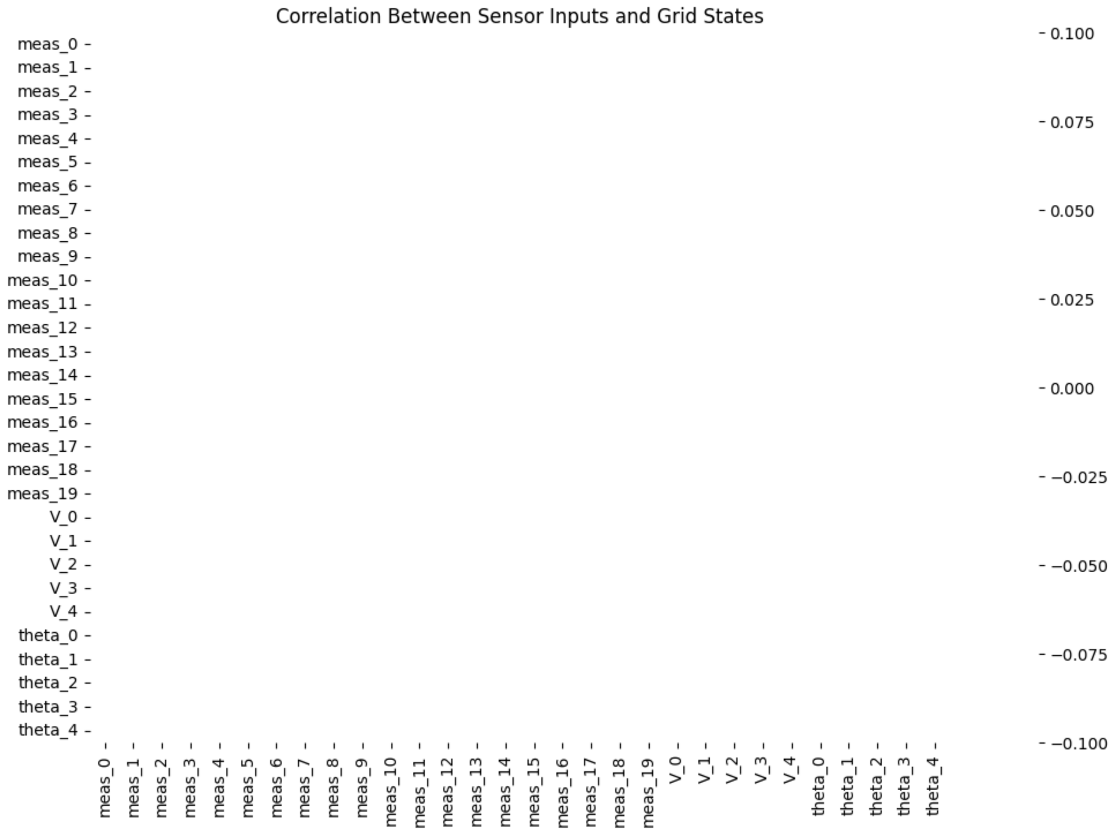
| | meas_8 | meas_9 | … | V_0 | V_1 | V_2 | V_3 | V_4 | theta_0 | theta_1 | \ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| meas_0 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| meas_1 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| meas_2 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| meas_3 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| meas_4 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| meas_5 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| meas_6 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| meas_7 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| meas_8 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| meas_9 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| meas_10 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| meas_11 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| meas_12 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| meas_13 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| meas_14 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| meas_15 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| meas_16 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| meas_17 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| meas_18 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| meas_19 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| V_0 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| V_1 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| V_2 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| V_3 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| V_4 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| theta_0 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| theta_1 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| theta_2 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| theta_3 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| theta_4 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |

```
          theta_2  theta_3  theta_4
meas_0        NaN      NaN      NaN
meas_1        NaN      NaN      NaN
meas_2        NaN      NaN      NaN
meas_3        NaN      NaN      NaN
meas_4        NaN      NaN      NaN
meas_5        NaN      NaN      NaN
meas_6        NaN      NaN      NaN
meas_7        NaN      NaN      NaN
meas_8        NaN      NaN      NaN
meas_9        NaN      NaN      NaN
meas_10       NaN      NaN      NaN
meas_11       NaN      NaN      NaN
meas_12       NaN      NaN      NaN
meas_13       NaN      NaN      NaN
meas_14       NaN      NaN      NaN
meas_15       NaN      NaN      NaN
meas_16       NaN      NaN      NaN
meas_17       NaN      NaN      NaN
meas_18       NaN      NaN      NaN
meas_19       NaN      NaN      NaN
V_0           NaN      NaN      NaN
V_1           NaN      NaN      NaN
V_2           NaN      NaN      NaN
V_3           NaN      NaN      NaN
V_4           NaN      NaN      NaN
theta_0       NaN      NaN      NaN
theta_1       NaN      NaN      NaN
theta_2       NaN      NaN      NaN
theta_3       NaN      NaN      NaN
theta_4       NaN      NaN      NaN

[30 rows x 30 columns]
```

[20]:
```python
# Plot heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=False, cmap='coolwarm', center=0)
plt.title('Correlation Between Sensor Inputs and Grid States')
plt.savefig('correlation_heatmap.png')
plt.show()
```

```
/Users/abhinavjha/Drive/DTU Project/.venv/lib/python3.12/site-
packages/seaborn/matrix.py:202: RuntimeWarning: All-NaN slice encountered
  vmin = np.nanmin(calc_data)
/Users/abhinavjha/Drive/DTU Project/.venv/lib/python3.12/site-
packages/seaborn/matrix.py:207: RuntimeWarning: All-NaN slice encountered
  vmax = np.nanmax(calc_data)
```




[21]:
```python
# Print correlations for V_0
print("Correlations with V_0:")
print(corr_matrix['V_0'].filter(like='meas_').sort_values(ascending=False).
    ↪head())
```

```
Correlations with V_0:
meas_0   NaN
meas_1   NaN
meas_2   NaN
meas_3   NaN
meas_4   NaN
Name: V_0, dtype: float64
```