Strategies for Large-Scale Renewable Integration in Power Systems: A Focus on Sensitive Bus Analysis

Mir Sayed Shah Danish, *Senior Member, IEEE*, Soichiro Ueda, *Student Member, IEEE* and Tomonobu Senjyu, *Fellow, IEEE*

Appendix

Draft Version for Review.

```
"""
Copyright © 2023, Danish M.
This work is licensed under a Creative Commons Attribution 4.0 International
License. This license allows others to distribute, remix, adapt, and build
upon this work, even commercially, as long as they credit the original
creation.
For more information about this license, please visit
https://creativecommons.org/licenses/by/4.0/
For any use beyond the scope of this license, please contact the copyright
holder at danish@mdanish.me.
"""

# This script is designed to be executed within the Jupyter Lab environment,
a web-based interactive development interface used for Python programming.
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
import warnings

# The script loads the dataset corresponding to the first feeder's 15 load
bus data from the Iowa 240 bus test system.
# The feeder 1 dataset can be found from this url:
https://github.com/mirsayedshah/230704.git
df = pd.read_csv('feeder1_load_data.csv')

# Step 0: Sensitivity analysis
# Calculate mean and standard deviation for each bus.
mean_loads = df.mean()
std_devs = df.std()

# Calculate normalized standard deviation for each bus.
norm_std_devs = std_devs / mean_loads

# Print out the buses with the highest normalized standard deviations. These
are the buses that are most sensitive to load changes over time.
most_sensitive_buses = norm_std_devs.sort_values(ascending=False)
print(most_sensitive_buses)

# Step 1: Data Preparation
# No specific data preparation steps are considered for this case study.

# Step 2: Correlation Analysis
# Select numeric columns from the dataframe.
numeric_columns = df.select_dtypes(include=[float, int]).columns
# Calculate correlation matrix.
correlation_matrix = df[numeric_columns].corr()
```

```python
# Visualize correlation matrix using a heatmap.
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.savefig('correlation_matrix.png')
plt.show()
plt.close()

# Export correlation matrix to CSV file.
correlation_matrix.to_csv('correlation_matrix.csv', index=True)

# Step 3: Regression Analysis
# Prepare the target variable and explanatory variables.
X = df[['B01', 'B02', 'B04', 'B05', 'B06', 'B07', 'B08', 'B09', 'B10', 'B11',
'B12', 'B13', 'B14', 'B15']]
y = df['B03']

# Add a constant term to the explanatory variables.
X = sm.add_constant(X)

# Fit the regression model.
model = sm.OLS(y, X)
results = model.fit()

# Print regression results summary.
print(results.summary())

# Export regression results to CSV file.
results_summary = results.summary().tables[1].as_csv()
with open('regression_results.csv', 'w') as f:
    f.write(results_summary)

# Visualization of regression results.
plt.figure(figsize=(10, 6))
plt.scatter(y, results.fittedvalues)
plt.xlabel('Actual B03')
plt.ylabel('Predicted B03')
plt.title('Regression Results - Actual vs Predicted B03')
plt.savefig('regression_results.png')
plt.show()
plt.close()

# Create a DataFrame with actual and predicted values.
regression_results_df = pd.DataFrame({
    'Actual B03': y,
    'Predicted B03': results.fittedvalues
})

# Export regression results (actual vs predicted B03) to CSV.
regression_results_df.to_csv('regression_results_actual_vs_predicted.csv',
index=False)

# Step 4: Granger Causality Analysis
# Perform the Granger causality test.
with warnings.catch_warnings():
    warnings.simplefilter("ignore", category=FutureWarning)
    granger_results = sm.tsa.stattools.grangercausalitytests(df[['B03',
'B01']], maxlag=3, verbose=False)

# Export Granger causality results to CSV file.
granger_results_summary = ""
for lag in granger_results.keys():
    result = granger_results[lag][0]['ssr_chi2test']
    result_summary = f"Lag: {lag}\n"
    result_summary += f"F-statistic: {result[0]}\n"
    result_summary += f"p-value: {result[1]}\n\n"
    granger_results_summary += result_summary
```

```python
with open('granger_causality_results.csv', 'w') as f:
    f.write(granger_results_summary)

# Print Granger causality results.
for lag in granger_results.keys():
    print(f'Lag: {lag}')
    print(granger_results[lag])

# Visualization of Granger causality results.
lag_values = []
p_values = []

for lag in granger_results.keys():
    lag_values.append(lag)
    p_values.append(granger_results[lag][0]['ssr_chi2test'][1])

plt.figure(figsize=(10, 6))
plt.plot(lag_values, p_values, marker='o')
plt.xlabel('Lag')
plt.ylabel('p-value')
plt.title('Granger Causality Results')
plt.xticks(lag_values)
plt.axhline(y=0.05, color='red', linestyle='--', label='Significance
Threshold (p=0.05)')
plt.legend()
plt.savefig('granger_causality_results.png')
plt.show()
plt.close()


# This script is designed to be flexible and adaptable. Additional
visualizations, tables, calculations, and summaries can be incorporated as
needed to present the results. These elements can be tailored and modified
based on specific requirements, ensuring a comprehensive and customized data
analysis outcome.

# ...

# End of the script

"""
Author: Mir Sayed Shah Danish
Date: July 4, 2023 (Last updated)

This script was developed for educational and informational purposes only.

Acknowledgments:
The author expresses his sincere gratitude to those who have added value
through their insightful feedback and significant contributions to this
project.

For any questions or further information, please contact danish@mdanish.me.
"""
```