The University of Manchester

Modelling the future GB power system

A dissertation submitted to The University of Manchester for the degree of

Master of Science in Electrical Power Systems Engineering

in the Faculty of Science and Engineering

Year of submission

2025

Student ID

11610796

School of Engineering

Contents

Word count: 16516

List of Figures

List of Tables

List of Abbreviations

- **AC** – Alternating Current
- **CCGT** – Combined Cycle Gas Turbine
- **DC** – Direct Current
- **DCOPF** – Direct Current Optimal Power Flow
- **ED** – Economic Dispatch
- **ESO / NGESO / NESO** – National Energy System Operator (formerly National Grid Electricity System Operator)
- **ETAP** – Electrical Transient Analyzer Program
- **ETYS** – Electricity Ten Year Statement
- **FES** – Future Energy Scenarios
- **GW** – Gigawatt
- **kV** – Kilovolt
- **LLM** – Large Language Model
- **LCOE** – Levelised Cost of Electricity
- **MATPOWER** – MATLAB Power System Simulation Package
- **MW** – Megawatt
- **MVA** – Megavolt-Ampere
- **N-1** – Security criterion meaning the system can withstand the loss of one element (e.g., line, generator, or transformer) without loss of supply
- **NGET** – National Grid Electricity Transmission (England & Wales Transmission Owner)
- **OFTO** – Offshore Transmission Owner
- **OPF** – Optimal Power Flow
- **pp / pandapower** – Python package for power system modelling and analysis
- **PSS®E** – Power System Simulator for Engineering (Siemens PTI)
- **PU** – Per Unit (normalized value system for power systems)
- **SHET** – Scottish Hydro Electric Transmission (Transmission Owner in northern Scotland)
- **SPT** – Scottish Power Transmission (Transmission Owner in southern Scotland)

Abstract

The transition to a low-carbon electricity system is reshaping the Great Britain (GB) transmission network, with growing renewable capacity, changing demand patterns, and increased power transfers across the country. To study these developments, this dissertation develops an automated modelling framework that builds a detailed bus-level representation of the National Grid Electricity Transmission (NGET) system directly from public datasets. The model is implemented in Python using pandapower and incorporates a DC Optimal Power Flow (DC OPF) with cost functions based on the Levelised Cost of Electricity (LCOE) for different generation technologies.

The framework is validated against trusted reference data and then applied to representative studies. Results show that the model reproduces realistic dispatch patterns and captures key operational constraints such as congestion on the north–south transfer boundaries. Additional scenarios demonstrate how changes in renewable generation and interconnector flows alter network stress, highlighting the usefulness of the approach for exploring future operating conditions.

This work provides a reproducible and extensible platform for analysing the GB transmission system. By linking open data with automated modelling, it creates a tool that can support research, scenario analysis, and long-term planning in the evolving electricity sector.

Declaration of originality

I hereby confirm that this dissertation is my own original work unless referenced clearly to the contrary, and that no portion of the work referred to in the dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Intellectual property statement

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the "Copyright") and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made **only** in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the "Intellectual Property") and any reproductions of copyright works in the thesis, for example graphs and tables ("Reproductions"), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=24420), in any relevant Dissertation restriction declarations deposited in the University Library, and The University Library's regulations (see http://www.library.manchester.ac.uk/about/regulations/_files/Library-regulations.pdf).

Acknowledgements

I would like to sincerely thank my supervisor, Dr. Robin Preece, for his continuous support throughout this project. His clear reasoning, constructive suggestions, and provision of resources have been instrumental in shaping both the direction and quality of my work.

My special thanks also go to Yitian Dai, whose assistance in developing the model and sharing additional insights greatly improved the outcomes of this study.

Above all, I am deeply grateful to my family for their constant encouragement, patience, and belief in me. Their support has been the foundation that made this work possible.

.

1 Introduction

This chapter introduces the background, context, and purpose of the project, highlighting the challenges facing the Great Britain (GB) power system during the energy transition. Rapid changes in generation and demand, alongside continuous investment in transmission and distribution, mean the system is constantly evolving. Against this backdrop, the chapter outlines the motivation for an automated, adaptable modelling approach, presents the project's aim and objectives, and defines its scope and intended outcomes, providing the foundation for the methodology and analysis in later chapters.

1.1 Background

The Great Britain (GB) electricity transmission network is the backbone of the national power system. It carries electricity from large power stations and renewable generation sites to the regional distribution networks that supply homes and businesses. The onshore network is managed by three Transmission Owners (TOs): National Grid Electricity Transmission (NGET) in England and Wales, SP Transmission (SPT) in southern Scotland, and Scottish Hydro Electric Transmission (SHE Transmission) in northern Scotland. Since 2019, overall coordination has been carried out by the National Energy System Operator (NESO), which is responsible for balancing supply and demand and keeping the system secure [1]. Offshore Transmission Owners (OFTOs) also play an important role by connecting offshore renewable generation, especially wind farms, to the onshore grid.

In recent years, the GB power system has been changing quickly. The retirement of coal plants has almost completely removed coal from electricity generation [2]. At the same time, renewable energy sources such as offshore wind and solar power have grown strongly. For example, renewables supplied around 44% of the UK's electricity in 2023 compared with only 7% in 2010 [3]. These changes have created new challenges, including more variable generation, new patterns of power flows, and the need for greater flexibility in the system [4].

To support this transition, there has been very high investment in the transmission network. One example is a £58 billion programme to expand capacity and build new high-voltage routes that will carry electricity from areas of high renewable generation in Scotland to centres of demand in England [5]. The energy regulator, Ofgem, has also approved £4 billion of investment to speed up grid upgrades and help meet the government’s clean energy targets for 2030 [6]. These investments show that the GB network is not static but is constantly developing in response to new requirements.

This shows that the GB power system is undergoing continuous change, shaped by decarbonisation, decentralised generation, and large levels of network investment. Because of this, there is a clear need for approaches that can represent the system in a way that captures both its current state and how it is evolving. Such approaches provide the basis for future energy scenario analysis and long-term planning.

1.2 Motivation

The Future Energy Scenarios (FES), published annually by NESO, outline possible pathways for how the GB power system may evolve under different policies, technologies, and consumer behaviours [7].They are not forecasts but structured scenarios that are widely used in policy, planning, and academic work to explore long-term challenges [8], [9]. The datasets include projections of generation capacity, regional demand, and electrification, making them a valuable foundation for testing future system conditions. To make effective use of these scenarios, detailed and adaptable models are needed that can capture power flows, identify bottlenecks, and evaluate system behaviour under varying assumptions. As highlighted in the FES reports [7], the increasing complexity of the GB system means modelling approaches must be capable of testing multiple scenarios efficiently. This study is motivated by that need and aims to develop an automated and reproducible transmission model that links directly to public datasets and provides a framework for scenario analysis and security assessment.

1.3 Need for Automated GB Transmission Modelling

While there is extensive literature on power system modelling, very few studies focus on the automation of model creation for the Great Britain (GB) transmission network. Existing research on GB often relies on proprietary models or reduced-order representations. For example, Lyden *et al.* [8], [10] introduced PyPSA-GB, which provides a 29-bus and zonal representation for scenario studies. Such models are valuable for high-level planning and policy analysis but do not capture the full network detail of the GB transmission system. This leaves a gap for automated, high-resolution modelling that can link directly to large datasets. Synthetic test systems such as those developed by Birchfield et al. [11] demonstrate how large, openly available transmission models can be constructed for research and benchmarking purposes. However, these are generic in design and do not reflect the specific characteristics of the GB system. Other studies use Future Energy Scenarios (FES) data to explore flexibility or renewable integration [12], but these rely on energy flow models that assume the network can always deliver power, representing only boundary constraints, rather than automating public datasets into detailed transmission models.

Although data and modelling tools for the GB transmission system exist, building a detailed and reproducible model remains challenging. Most available approaches are either proprietary, manually constructed, or simplified, which limits their adaptability and scalability. What is missing is a framework that can automatically translate large public datasets into a consistent, detailed model suitable for scenario analysis.

1.4 Aims and Objectives

The aim of this study was to develop and validate an automated model of the Great Britain transmission network using an open-source tools and public data, to enable scenario analysis and network security assessment.

The main objectives of this study were:

1. To review and select a suitable modelling approach and tool for power flow and transmission network studies.
2. To collect, extract, and process transmission network data, including lines, transformers, generation, and demand.
3. To construct and automate the creation of a GB transmission system model using the selected tool.
4. To validate the model against published system characteristics or reference data.
5. To use the model to identify transmission line constraints, assess Future Energy Scenario (FES) and conditions where N-1 security is not maintained.
6. To produce documentation that ensures the model can be reused, reproduced, and extended in future work.

1.5 Contributions

All core objectives were achieved: a suitable modelling approach was selected, NESO datasets were processed into consistent network elements, and an automated workflow was developed to construct the NGET model. The model was validated against reference data, showing accurate replication of dispatch patterns and boundary constraints, and was applied to scenario analysis and an N-1 study.

The only partial objective was the application to Future Energy Scenarios: while a simplified scenario was considered to demonstrate applicability, a full, extensive FES study was beyond the scope of this work.

1.6 Scope and Limitations

This study focuses developing an automated model of the Great Britain transmission network using an appropriate power flow approach. The work is limited to steady-state analysis and does not include detailed voltage behaviour, dynamic system responses, or integration with real-time control systems.

1.7 Thesis structure

This thesis is comprised of five main body chapters.

Chapter 1 introduces the study. It explains why modelling the GB transmission network is important, outlines the aim and objectives, and defines the scope and limitations. The structure of the thesis is also presented.

Chapter 2 reviews and evaluates different modelling approaches and tools for automating the GB transmission network. It considers their strengths and limitations, before identifying the most suitable option. The chapter also discusses existing studies that have applied transmission system models and scenario-based analysis to the GB network.

Chapter 3 sets out the methodology. It explains how the modelling framework was developed, including the use of public datasets, the automated creation of the network, and the steps taken to reproduce and validate the model. The procedures for scenario exploration and basic security studies are also outlined.

Chapter 4 presents the results. The emphasis is on validation, showing that the model accurately reproduces network behaviour, with selected analyses included to demonstrate its capability for scenario-based studies when required.

Chapter 5 concludes the thesis. It summarises the key findings, reflects on the effectiveness and limitations of the modelling framework, and places the work in the wider context of energy system research. The chapter also outlines directions for future extensions and opportunities for collaboration with industry and academia.

2 Background Research

This chapter reviews the literature relevant to automating the creation of a model of the GB transmission network. It introduces key concepts in power system modelling, compares different approaches and tools, and considers their strengths and limitations. The chapter also discusses previous studies on transmission system modelling and scenario analysis, highlighting how they inform the approach taken in this study.

2.1 The Great Britain Transmission System

The Great Britain (GB) transmission network is among the most extensive and technically demanding in Europe. It spans long geographical distances, interconnecting diverse regional demand centres with major generation hubs. Much of the system is characterised by strong regional imbalances: renewable generation, particularly offshore wind, is concentrated in Scotland and along the east coast, while the highest demand is centred in the Midlands and South of England. This creates substantial north–south power transfers across the system [13].

The rapid expansion of renewable generation has amplified these flows, adding variability and uncertainty to system operation. Offshore wind, now the largest single source of new capacity, can fluctuate significantly over short timescales, while solar power introduces its own diurnal and seasonal patterns. These resources are often located far from demand centres, placing additional strain on transmission corridors. As a result, congestion has become a recurring feature of GB system operation, leading to redispatch actions and high curtailment costs [14]. In 2024, the UK incurred over £1 billion annually in wasted wind generation due to grid constraints [15]. Additionally, the system operator's balancing costs rose by 10% in FY2024/25, reaching £2.7 billion, with £1.7 billion attributed specifically to thermal constraint management [16].

Alongside congestion, the reduction in synchronous generation poses new operational challenges. With coal plants retired and gas generation operating more flexibly, system inertia has fallen, making the network more sensitive to disturbances. Maintaining stability and security of supply in this context requires careful planning and a clear understanding of how the system will evolve [17].

Robust modelling of the GB transmission system is essential for assessing adequacy, understanding the impacts of new generation, and exploring future scenarios. It plays a central role in long-term planning and in meeting GB's decarbonisation and security goals.

2.2 Power System Modelling Approaches

Power system modelling covers a range of timescales, but at steady state, power flow analysis is the fundamental method for understanding grid behaviour and supporting planning and operation.

2.2.1 Power Flow Analysis

Power flow analysis is a central tool in power system studies because it describes how electricity moves across the network under steady-state conditions. By solving the network equations that govern bus voltages, phase angles, and line flows, it provides a snapshot of how generation and demand interact across the grid [18], [19]. This makes it indispensable in identifying congested transmission corridors and evaluating how renewable generation is integrated into the system. When generation in one region cannot be fully transmitted to demand centres, power flow analysis highlights the constraints that lead to redispatch actions and renewable curtailment, both of which are increasingly important in the GB system as renewable penetration rises.

2.2.2 AC and DC Power Flow

Two main formulations are widely applied: AC and DC power flow. AC power flow solves the full set of nonlinear equations for active and reactive power, capturing voltage magnitudes, reactive power behaviour, and losses. This makes it essential for detailed studies of voltage stability and reactive support. However, solving these nonlinear equations for large systems such as GB is computationally intensive and can be difficult to scale.

DC power flow, by contrast, applies simplifying assumptions: voltages are fixed near 1.0 pu, resistances are neglected, and angle differences are assumed small [20]. This reduces the problem to a linear set of equations that approximate active power transfers [21]. While DC power flow cannot capture voltage behaviour or losses, it is efficient and scalable, making it well suited for large-scale scenario analysis, congestion studies, and renewable integration assessments. Since this study is concerned primarily with transmission constraints rather than voltage management, DC power flow is therefore selected as the more appropriate method, as it captures the active power flows of the network while requiring far less computation, making it suitable for large interconnected systems.

2.2.3 Optimisation-based Approach

The drawback of standard power flow methods is that they depend on a slack bus to absorb mismatches and ignore operating constraints such as generator limits and thermal line ratings, meaning they do not fully capture real-world conditions [19]. To overcome these shortcomings, optimisation-based approaches extend the power flow framework to include both physical limits and operational objectives.

The simplest of these approaches is Economic Dispatch (ED), which determines the least-cost generator outputs required to meet demand. ED considers generator cost functions and operating limits but does not account for the physics of the transmission network. As a result, a solution that appears economical in theory may be infeasible in practice, since power cannot always be delivered across congested or capacity-limited lines [21].

Optimal Power Flow (OPF) extends standard power flow by incorporating both network physics and economic objectives, producing dispatch schedules that are feasible as well as least-cost [22], [23]. Two main variants are commonly applied. AC OPF captures the full nonlinear behaviour of the system, but it is computationally intensive and can struggle with convergence in large or stressed networks [24]. DC OPF, by contrast, simplifies the problem to a linear form, solvable as a convex optimisation problem, which is suitable for large-scale studies [20].

2.3 Power System Modelling Tools

Power system studies rely on specialised software to apply power flow and optimisation methods to real networks. These tools implement the numerical solvers, handle system data, and provide the framework for analysing network behaviour. Choosing the right tool is therefore essential to ensure that the modelling approach matches the objectives of the study.

2.3.1 General Modelling Environments

A wide range of software platforms exist for power system studies, supporting applications from steady-state analysis to dynamic and transient simulations. Widely used examples include PSS®E, which has long been applied to transmission planning and power flow studies using Newton–Raphson and fast-decoupled solvers [25]. DIgSILENT PowerFactory extends beyond power flow into dynamic and stability analysis, with modules for protection coordination and electromagnetic transients [26]. ETAP integrates generation, industrial, and monitoring capabilities within a single environment, while PSCAD/EMTDC specialises in electromagnetic transient studies at very short timescales [27].

Most commercial platforms implement both AC and DC power flow, and many also provide optimal power flow (OPF) functions. These implementations are primarily designed for operational decision support, typically emphasising AC OPF with limited flexibility to modify formulations. While reliable for routine analysis, they are less suited to research applications where repeated OPF runs and customised formulations are required.

For this study, a critical requirement is the ability to automate network creation and analysis. Commercial platforms often rely on graphical interfaces and manual input of network data, which becomes inefficient and error-prone when scaling to large datasets or many scenarios. Scripting-based environments that support automated model generation and batch processing are therefore more suitable, as they allow networks to be constructed and solved consistently across hundreds of scenarios.

2.3.2 Tools for Optimal Power Flow

Given the need for automation and repeated evaluations, tools developed in research contexts are particularly relevant for optimal power flow studies. Among these, MATPOWER and pandapower are the most widely used and form the basis of much current academic and applied work.

MATPOWER is a MATLAB-based package developed specifically for steady-state operations, planning, and OPF research [28]. It implements AC and DC OPF formulations using Newton-based solvers for nonlinear problems and linear programming for DC cases. Its strength lies in being transparent and extensible, making it a benchmark platform for testing new OPF algorithms. However, its dependence on MATLAB, a proprietary environment, limits scalability and integration with modern data handling workflows. Running large numbers of scenarios or coupling OPF with external optimisation routines can become cumbersome, especially when compared with modern scripting environments.

Pandapower was developed to address these limitations by providing a Python-based framework for power system analysis [29]. Like MATPOWER, it supports AC and DC OPF, but its design emphasises automation and integration with data science tools. Networks can be created directly from tabular datasets using the Pandas data structure, which is particularly valuable when handling large interconnected systems. The framework supports repeated OPF evaluations with minimal manual intervention, enabling efficient large-scale scenario analysis. Pandapower also integrates with widely used optimisation solvers: linear programming solvers for DC OPF, and nonlinear solvers such as IPOPT (Interior Point OPTimizer) for AC OPF, giving it both flexibility and scalability [29].

In summary, MATPOWER remains a reference platform for OPF research but is constrained by its reliance on MATLAB. Pandapower extends the same philosophy into Python, offering a modern, data-driven framework that supports automated model creation, scripting, and large-scale analysis. These features make it particularly suitable for studies such as this.

2.3.3 Selection of Modelling Tool and Approach

This study adopts DC OPF implemented in pandapower. DC OPF is chosen because its linear formulation allows fast, stable solutions for large interconnected systems, making it well suited to analysing power transfers and congestion in the GB transmission network. Pandapower is selected as the tool because it enables automated model creation directly from datasets, integrates smoothly with Python for data handling, and supports consistent scenario-based analysis.

2.4 Critical Comparison of Previous Work on GB Modelling

Although power system modelling has a long history, relatively few studies have focused on openly available, high-resolution modelling of the Great Britain (GB) transmission system. Existing work tends to fall into three broad categories: simplified zonal models, detailed but proprietary models, and synthetic test systems.

One of the most notable open-source contributions is PyPSA-GB, developed by Lyden et al. [8], [30] which provides a 29-bus zonal representation of the GB system. Its strength lies in accessibility and ease of use for policy and scenario studies, but its zonal structure inevitably masks detailed network constraints, such as congestion on specific transmission corridors. The model is therefore more appropriate for long-term strategic planning than for operationally relevant analysis.

Imperial College London and other academic institutions have developed zonal models of GB transmission for specific studies, often focusing on renewable integration and stability [31]. These models typically use data from the National Grid ESO (NGESO) and employ AC power flow or OPF methods [31]. However, they are usually developed for one-off studies, remain proprietary, and are not released in a form that can be reproduced or automated by others [32].

The NGESO itself maintains proprietary planning and operational models, which are the most detailed representations of the GB system [33], [34]. While these form the basis of network planning decisions, they are not openly available and require commercial licences to access [35]. As such, they cannot serve as a basis for reproducible academic research.

A third category is represented by synthetic systems, such as those of Birchfield et al. [36]-[44], which construct large test networks with realistic statistical properties. These are openly available and valuable for benchmarking algorithms, but they are generic in design and do not reflect the specific characteristics of the GB system, such as strong north–south flows or the rapid expansion of offshore wind.

Table 2.1: High-level comparison of representative GB models:

Model / Source	**Resolution**	**Openness & Data Basis**	**Reproducibility**	**FES Scenario Use**
PyPSA-GB [8]	Zonal (29-bus)	Open-source, ESO/FES public	Reproducible, manual updates	Good for policy, not detailed flows
Imperial models [31]	Zonal	Proprietary, ESO data	Limited, one-off	Scenario-focused, not repeatable
NGESO tools [35]	Full bus-level	Proprietary, internal ESO	Not reproducible	Official planning, not research
Synthetic systems[36]-[44]	Bus-level (generic)	Open-source, synthetic	Fully reproducible	Benchmarking only, not GB-specific
This study	Bus-level (GB-specific)	Open-source, ESO appendices	Automated, fully reproducible	GB-specific, high-resolution analysis

As shown in Table 2.1, each existing approach to GB transmission modelling has strengths but also clear limitations. PyPSA-GB is open and reproducible but restricted to a 29-bus zonal representation, limiting detailed flow analysis. Imperial College models use ESO data but remain proprietary, developed for one-off studies that cannot be repeated or extended. NGESO's internal tools are the most accurate bus-level models but are inaccessible outside the operator. Synthetic systems are openly available and reproducible but generic, lacking GB-specific characteristics.

This study addresses that gap by developing a bus-level GB model directly from ESO's public datasets. Like PyPSA-GB, it is open and reproducible, but unlike zonal or synthetic systems it captures detailed GB flows, while avoiding the inaccessibility of proprietary models. The novelty lies in combining automation with high resolution, enabling repeated, consistent scenario analysis.

2.5 Summary

This chapter reviewed the literature on GB transmission system modelling and the case for automation. It highlighted the system's main challenges, including regional imbalances driving north–south transfers, increasing congestion and curtailment from renewable growth. Different modelling approaches were considered, from power flow methods to OPF, with DC OPF identified as the most appropriate formulation for this study. Among available tools, pandapower was selected for its ability to automate network creation and support reproducible scenario analysis. In doing so, the chapter fulfils Objective 1: to review and select a suitable modelling approach and tool for power flow and transmission network studies.

3 Methodology

This chapter presents the methodology for developing a reproducible, high-resolution model of the Great Britain (GB) transmission network using Python and pandapower. The approach is code-based rather than graphical, with snippets and pseudo-code illustrating the workflow. Additional libraries, notably pandas, support data handling. Snippets shown are simplified; the full implementation is documented separately..

3.1 Methodological Framework

The purpose of this methodology is to produce an automated model of the National Grid Electricity Transmission (NGET) portion of the GB transmission system, which can later be scaled to cover the full network. At a high level, the framework:

- Takes structured datasets as inputs.
- Produces a pandapower network object as output.
- Is modular and adaptable for updates or scenario variations.
- Demonstrates reproducibility: model can be recreated from the source files.

The outputs of this chapter are:

1. A functioning bus-level model of the NGET transmission system.
2. A clear strategy for validation.
3. A framework that demonstrates applicability to scenario exploration.

3.2 Methodological Workflow

This section describes the process of data preparation, cleaning, network model creation, and validation. It provides a structured overview of the steps undertaken to ensure an accurate and reliable network model.

3.2.1 Data-Driven DC OPF Modelling

The primary data source is the NESO Data Portal, particularly the ETYS appendices, which provide generation, demand, and network data. Table 3.1 summarises the relevant appendices and their contents.

Table 3.1: ETYS Appendices and Data Used

Appendix	Content	Key Data Provided
B	Network Data	Substation codes, transmission circuits (lines, cables, series devices), transformers (shunts ignored for DC studies)
F	Generation Data	Connection sites, project names, installed capacity (MW), project status (built/unbuilt), plant type
G	Demand Data	Node identifiers (bus IDs) and corresponding MW demand values

To represent interconnections, the NESO Interconnector Register [45] is also used, as these assets are essential for validation of cross-border flows. Together, these datasets provide the necessary foundation for constructing a bus-level model of the GB transmission network.

3.2.2 Data Preparation

The ETYS appendices provide the core datasets for this study. Since the methodology is programming-based, the structure of each appendix is important, as column headings determine how data are extracted in pandas (data extraction tool).

ETYS Appendix B – Network Data

Includes substations, transmission circuits, and transformers data sheets (shunt data ignored as not relevant for DC). Examples are provided in Tables 3.2 to 3.4.

Table 3.2: Transmission Circuit Data

Node 1	Node 2	OHL (km)	Cable (km)	Circuit Type	X (% on 100 MVA)	Winter Rating (MVA)	Summer Rating (MVA)
ABHA4A	EXET41	48.785	0.000	OHL	0.9831	2078	2078
WTHU4C	WTHU4E	0.000	0.000	Series Reactor	1.0000	2400	2400
BIRK21	LISD2A	0.000	12.891	Cable	0.2978	817	817

Table 3.3: Transformer Data

Node 1	Node 2	X (% on 100MVA)	Rating (MVA)
ABHA4A	ABHA11	8.4025	275

Table 3.4: Substation Data

Site Code	Site Name	Voltage (kV)
ABBA	ABERDEEN BAY WINDFARM	132

ETYS Appendix G – Load Data

Contains a single sheet listing bus IDs and their demand values in MW, with an example shown below in table 3.5.

Table 3.5: Load Data

Node	24/25 MW
ABHA4A	41

ETYS Appendix F – Generation Data

Provides generator connection information, including site, capacity, project status, and plant type (important for cost and scenario analysis). An example is given with table 3.6.

Table 3.6: Generation Data

Project Name	Connection Site	MW Connected	Project Status	HOST TO	Plant Type
17 Acres and BESS	Tealing 275/33kV Substation	0.00	Scoping	SHET	Energy Storage System
A'Chruach Wind Farm	A'Chruach Wind Farm 275kV Substation	43.00	Built	SHET	Wind Onshore
Abedare	Upperboat 132kV Substation	10.00	Built	NGET	CCGT

All data must be structured this way for automated modelling.

3.2.3 Data Cleaning and Extraction

Once the datasets are identified and structured, the next stage is to prepare them for use in pandapower. This involves extracting the relevant sheets, cleaning the data, and handling missing data to ensure consistency across files.

3.2.3.1 Extraction with pandas

Using pandas, Excel and CSV files can be imported directly into Python as dataframes, as shown in Figure 3.2 where a transformer rating is extracted.

```
# Import the pandas library
import pandas as pd

# Define the file and sheet to be read
document_name = "ETYS_B.xlsx"
sheet_name = "Transformers"

# Read the selected sheet into a dataframe
df_trafo = pd.read_excel(document_name, sheet_name)

# Columns can be accessed directly, e.g., transformer ratings:
df_trafo["Rating"]
>>> 400      # transformer rating = 400 MVA
```

Figure 3.1: Loading Transformer Rating from Excel with Pandas

3.2.3.2 Standard Cleaning Functions

Helper functions are used to harmonise units and prepare data for OPF analysis. All values are converted to a 100 MVA, 400 kV system base.

As pandapower requires line ratings in kA but ETYS provides them in MVA, a conversion function is applied. Figure 3.3 illustrates the conversion applied.

```
# This function converts MVA ratings to kA.
def convert_mva_to_ka(mva_rating):
    ka_rating = (mva_rating * 1000) / (math.sqrt(3) * 400)
    return ka_rating
# Example
convert_mva_to_ka(500)
>>> 721.69     # equivalent line rating in kA
```

Figure 3.2: Conversion of Line Ratings from MVA to kA

Similarly, a helper function converts reactances from p.u. to ohm/km for use in pandapower as shown in figure 3.4.

```
# Convert reactance from p.u. to ohm/km
def convert_x_from_pu_to_ohm_per_km(x_pu, length_km):
    z_base = (400e3 ** 2) / 100e6   # base impedance at 400 kV,
100 MVA
    x_ohm = (x_pu / 100) * z_base   # convert from percent to
ohms
    return x_ohm / length_km

# Example
convert_x_from_pu_to_ohm_per_km(3.7, 100)
>>> 0.562    # reactance in Ω/km
```

Figure 3.3: Conversion of Reactance from p.u. to Ω/km

Plant types are grouped into broader LCOE categories to ensure consistent cost modelling; the example shown in figure 3.5 is not exhaustive.

```
# mapping plant types for LCOE category
def map_to_lcoe_category(plant_type):
    if plant_type in ["Onshore Wind", "Offshore Wind"]:
        return "Wind"
    elif plant_type == "Solar PV":
        return "Solar"
    else:
        return "Other"

# Example
map_to_lcoe_category("Onshore Wind")
>>> 'Wind'
```

Figure 3.4: Mapping Plant Types to LCOE Categories

3.2.3.3 Handling Missing Nodes

Not all values are directly available in the raw appendices.

For load allocation, if demand is only given at the regional or substation level, it is distributed evenly across the associated buses. This ensures each bus receives a share of the total demand. The code snippet in figure 3.6 illustrates this process.

```
# Distribute regional/substation load evenly across buses
for substation, total_load in regional_load:
    number_of_buses = len(substation.buses)
    load_per_bus = total_load / number_of_buses

    for bus in substation.buses:
          pp.create_load(net, bus, p_mw=load_per_bus)
```

Figure 3.5: Distribute regional/substation load evenly across buses

Generator nodes are more complex to allocate. In cases where generator buses are not explicitly listed, connection sites are matched against substation names using fuzzy matching, requiring at least an 80% match. Once the substation is identified, the generator capacity is divided evenly among its buses. The pseudo-code below in figure 3.7 illustrates this process.

```
# Match generator connection sites to substations using fuzzy
matching
for generator in generators:
    substation = fuzzy_match(generator.site_name,
substation_names, threshold=80)

    if substation:
        number_of_buses = len(substation.buses)
        gen_per_bus = generator_capacity / number_of_buses

        for bus in buses:
            pp.create_gen(net, bus, p_mw=gen_per_bus)
```

Figure 3.6: Match generator connection sites to substations using fuzzy matching

This ensures that generators without explicit bus IDs are mapped to the most likely connection site, preserving network integrity.

3.2.4 Applying the Network Model

The model is designed to be modular, meaning changes such as adding new generators or updating line ratings can be made without altering the workflow. This modularity mirrors the dataset structure, where generation, demand, and network assets are provided in separate files. An example is illustrated in Figure 3.7 where the user loads the network, demand, and generator files along with their respective element sheets.

```
# --- User Inputs ---
network_file = "network_data.xlsx"
demand_file = "demand_data.xlsx"
generator_file = "gen_data.xlsx"

substation_sheet = "substations"
transformer_sheet = "transformers"
line_sheet = "lines"
load_sheet = "loads"
gen_sheet = "generators"
```

Figure 3.7: User Input files and element sheets

To build this network model, a dedicated script (`network.py`) was created. The script initialises an empty pandapower network and populates it with buses, lines, transformers, loads, and generators based on user-provided datasets. Once the inputs are defined, the script automates the creation of the full network model, which can then be analysed, as illustrated in Figure 3.8.

```
# --- Create Empty Network ---
net = pp.create_empty_network()

# --- Build Network Elements ---
```

```
create_buses(net, network_file, substation_sheet)
create_lines(net, network_file, line_sheet)
create_transformers(net, network_file, transformer_sheet)
create_loads(net, demand_file, load_sheet)
create_gens(net, generator_file, gen_sheet, substation_sheet)

# --- Export Results ---
pp.to_excel(net, "results/network_results.xlsx")   # network res
pp.to_excel(net, "results/dc_opf_results.xlsx")    # dc opf res
```
Figure 3.8: Network creation and export of results using the network.py script

Running the `network.py` script executes the full workflow.

The results are exported to Excel files: a network file (`network_results.xlsx`) and a DC OPF file (`dc_opf_results.xlsx`), which can then be used for further studies and analysis. The stepwise construction process is reflected in the following subsections, where each function is introduced in detail.

3.2.5 Creation of Network Elements

This section explains how each element of the network is created from the input datasets.

3.2.5.1 Buses

The *create_buses* function reads the substation table (codes, names, and nominal voltages), creates pandapower bus nodes, and stores a lookup (bus → id) for later use, as shown in Figure 3.9.

```
# create_buses(net, network_file, substation_sheet)
def create_buses(net, xlsx_path, sheet):
    v_base = 400_000
    df_sub = pd.read_excel(xlsx_path, sheet_name=sheet)
    bus_lookup = {}
    for _, bus in df_sub.iterrows():
        # Create buses
        bus_id = pp.create_bus(net, v_base, bus_name)
        bus_lookup["bus_substation_code"] = bus_id
    return bus_lookup
```
Figure 3.9: create_buses function

3.2.5.2 Lines

The *create_lines* function adds transmission circuits using electrical parameters and end-points. It converts ratings to kA where required, assigns X parameters, and preserves zero-length connections by using a small length, as shown in Figure 3.10.

```
def create_lines(net, file, sheet, bus_lookup):
    df_lines = pd.read_excel(file, sheet_name=sheet)

    for _, line in df_lines():
        from_bus = bus_lookup.get(line["from_bus"])
        to_bus   = bus_lookup.get(line["to_bus"])
```

```
        length = line["Length_km"] or 0.01  # Handle zero length
        x  =  convert_x_pu_to_ohm_per_km(x_pu, length_km)
        ka_rating = convert_mva_to_ka(line["mva_rating"])

        # Add line to pandapower
        pp.create_line_from_parameters(net, from_bus, to_bus,
length, x, ka_rating)
```

Figure 3.10: create_lines function

3.2.5.3 Transformers

The *create_transformers* function models transformers between buses using impedance elements, as shown in Figure 3.11.

```
# create_transformers(net, network_file, transformer_sheet)
def create_transformers(net, xlsx_path, sheet, bus_lookup):
    df_trafos = pd.read_excel(xlsx_path, sheet_name=sheet)

    for trafo in df_trafos:
        x_pu = trafo["x"] / 100
        mva_rating = trafo["rating_mva"]

        # Add transformer as impedance element
        pp.create_impedance(net, from_bus, to_bus, x_pu,
mva_rating)
```

Figure 3.11: create_transformers function

3.2.5.4 Loads

The *create_loads* function assigns demand to buses by reading bus-level demand in MW or distributing substation totals across buses, as shown in Figure 3.12.

```
# create_loads(net, demand_file, load_sheet)
def create_loads(net, xlsx_path, sheet, bus_lookup,
subs_to_buses):
    df_loads = pd.read_excel(xlsx_path, sheet_name=sheet)

    # Case A: demand per bus directly
    if bus_name in df_loads.columns:
        for loads in df_loads):
            bus = bus_lookup.get(bus_name
            if bus is not None:
                pp.create_load(net, bus, demand_mw)

    # Case B already described earlier – distribute substation
demand across buses
```

Figure 3.12: create_loads function

3.2.5.5 Generators

The *create_gens* function allocates generation by connection site and plant type. It matches sites to substations using fuzzy matching, splits capacity across buses at the matched substation, creates generator elements for DC OPF active power, and assigns each generator a cost function based on its LCOE category, as shown in Figure 3.13.

```
# create_gens(net, generator_file, gen_sheet, substation_sheet)
def create_gens(net, xlsx_path, sheet):
    df_gens = pd.read_excel(xlsx_path, sheet_name=sheet)
    for gen in df_gens ():
        # Substation identified via fuzzy matching (see earlier
section)
        # Map plant type to LCOE category (done earlier)

        for bus in buses:
            pp.create_gen(net, bus, p_mw, slack=False)
        # Assign linear cost function to generator
            pp.create_poly_cost(net, element=gen,
cp1_eur_per_mw=gen_cost)
```

Figure 3.13: create_gens function

To represent generation costs within the OPF, this study adopts the Levelised Cost of Electricity (LCOE), a widely used metric that expresses the average lifetime cost of electricity per unit of output, using capital, operational, and fuel costs [46], [47].

3.3 Validation

The model is validated to ensure it produces plausible results and to demonstrate that it can be applied to FES analysis, generating outputs that can support research and decision-making. The outcomes of this validation are presented and discussed in the Results and Discussion chapter of this dissertation

3.3.1 Validation Requirements

Validation requires a set of reference points against which the model outputs can be assessed. The requirements applied are:

- **Results Review:** Assess network and DC OPF outputs for plausibility.
- **Using Reference Data:** Compare dispatch and boundary flows with published values under comparable loading.
- **Scenario Applicability:** Demonstrate the model in base cases (e.g., winter/summer loading) and standard security studies (N-1).
- **High Resolution:** Use bus-level detail to identify fully loaded lines.

3.3.2 Implementation of Validation

To implement the validation strategy, two additional elements are introduced:

Interconnectors: Based on the NESO Interconnector Register [45], key links (e.g., IFA, Nemo, BritNed, NSL, EWIC) are modelled as short lines with ratings to capture transfer capacity. Their operating mode is set manually and excluded from DC OPF optimisation.

Scottish Power Transmission (SPT) Assets: A reduced Scottish network is represented by aggregating demand and generation into single loads and equivalent generators by plant type. These participate in the DC OPF, enabling comparison with published boundary flow data while keeping the focus on the NGET network. The validation workflow proceeds as follows:

3.3.2.1 Reference Data Comparison

The system load is scaled to match published operating conditions, ensuring that the test cases represent realistic system states.

The base cases correspond to summer and winter loading. Once the load adjustments are complete, the DC OPF is executed using *pp.rundcopp* to produce generator dispatch and line flow results. The resulting outputs from `res_gen` are then collated and compared with reference generation patterns from Grid Templar to confirm that the model produces plausible dispatch.

Boundary transfers were aggregated from the exported `res_line` results (`dcopf_results.xlsx`) and compared with ESO's published data, providing a system-level benchmark. Individual line loadings were then assessed, with lines reaching 100% thermal limits used to identify critical transmission corridors and the conditions under which they become binding.

3.3.2.2 Scenario Analysis

In the scenario analysis, the load is increased by +10% to represent increased electrification. This is applied to both winter cases, with equivalent line constraints set to reflect the seasonal thermal limits. Renewables are also increased by +10% to represent higher renewable energy penetration as shown in figure 3.3.

```
net.gen.loc[net.gen["name"].str.contains("Wind|Solar"), "p_mw"]
*= 1.10
```

Figure 3.14: Implementation of Renewable Growth Scenario (+10%)

The DC OPF is then executed, and the resulting dispatch and flows are compared against the winter base cases.

3.3.2.3 N-1 Security

For the N-1 analysis, the winter base case is used as reference. System load is increased in small steps; after each increment, an N-1 screen is run by opening one line at a time and re-solving the power flow. The case is deemed insecure once the contingency evaluation returns False, i.e., any post-contingency element reaches or exceeds its limit. When the check returns False, the system has reached its maximum secure demand for this test case. The implementation is shown below in figure 3.4.

```
step = 0.02  # +2% per step
while True:
    scale += step
    net.load["p_mw"] *= (1 + step)      # increment load
    res = pp.run_contingency(net, nminus1_cases)
    if not res["is_secure"].all():      # check security success
        break
```

Figure 3.15: Implementation of N-1 Security Criterion

This chapter demonstrates the applicability of the framework using selected scenarios based on FES-style inputs. A comprehensive exploration of all possible scenarios is beyond the current scope and is reserved for future work.

3.4 Summary

This chapter has developed a reproducible, high-resolution model of the GB transmission system using Python and pandapower. Structured datasets from NESO's ETYS appendices were collected, processed, and integrated to automate the creation of a bus-level NGET model, with transformers represented as impedances and generators assigned LCOE-based cost functions. The model was validated against published reference generation patterns and boundary flows to ensure plausibility. Scenario analysis was implemented by scaling winter base case to reflect demand growth and renewable penetration, while N-1 contingency analysis was used to identify maximum secure demand.

In doing so, the chapter has fulfilled the study's objectives:

2. Collecting and processing network data.
3. Constructing and automating the model.
6. Producing a documented workflow that is reusable and extendable for future applications.

4 Results and discussion

This section presents results for the GB transmission system, focusing on NGET. Scotland and offshore systems are simplified as aggregated loads, generators, and interconnectors to capture wider interactions while analysing the England–Wales backbone. For the model to be considered successful, it must:

- Accurately parse and generate network elements from the dataset.
- Assemble a network model from the parsed elements, capable of DCOPF.
- Run analyses and scenarios reliably, and
- Produce results that align with reference data.

Failure to meet any of these criteria would indicate that the model cannot be relied upon for system studies.

4.1 The Network Model

The network model was generated from the parsed NESO dataset, comprising buses, generators, loads, lines, and transformers.

4.1.1 Base Network Parameters

To verify the integrity of the dataset, the exported network parameters were compared directly with the original NESO appendix data. The aim was to confirm that all network components were correctly parsed.

The NESO appendix provides the reference values for network equipment, as shown in *Table 4.1* for a sample of transformers. For example, a 400 kV transformer rated at 275 MVA connects bus ABHA4A to bus ABHA11, with reactance given as 8.4025% on a 100 MVA base. A second transformer links ABHA4B to ABHA12, rated at 264 MVA with the same reactance value.

Table 4.1: NESO Appendix Transformer Data

Node 1	Node 2	R (% on 100MVA)	X (% on 100MVA)	Rating (MVA)
ABHA4A	ABHA11	0.1658	8.4025	275
ABHA4B	ABHA12	0.1658	8.4025	264

The corresponding transformer dataset exported from the model is presented in *Table 4.2*. Here, the *from_bus* and *to_bus* IDs are correctly assigned, the MVA ratings are preserved, and the reactance values (originally in %X) are accurately converted into per unit (p.u.).

Table 4.2: Parsed Transformer Data from Model

	from_bus	to_bus	xft_pu	xtf_pu	sn_mva	in_service
0	3	495	0.084025	0.084025	275	TRUE
1	6	496	0.084025	0.084025	264	TRUE

A similar validation was carried out for the bus dataset. The model export, shown in *Table 4.3*, lists the bus IDs and voltage levels, confirming that the correct nodal structure of the transmission system has been reproduced.

Table 4.3: Parsed Bus Data from Model

bus_ID	name	vn_kv	in_service	geo
3	ABHA4A	400	TRUE	null
6	ABHA4B	400	TRUE	null
495	ABHA11	400	TRUE	null
496	ABHA12	400	TRUE	null

Validating these datasets is vital, as errors such as misallocated transformers or bus IDs can distort flows, while consistency with NESO data ensures the DC OPF and scenario analyses can be trusted to reflect realistic system behaviour.

4.1.2 Model Representation

Running the program to generate the model and plotting the network topology produced the interconnected system shown in *Figure 4.1*. In this representation, loads are depicted in red, generators in blue, and transmission lines in grey. The plot does not use actual ETYS coordinates or line lengths but a force-directed layout, drawing connected buses closer and unconnected ones apart. Strongly connected areas cluster, and some lines curve to reduce overlap and improve clarity.

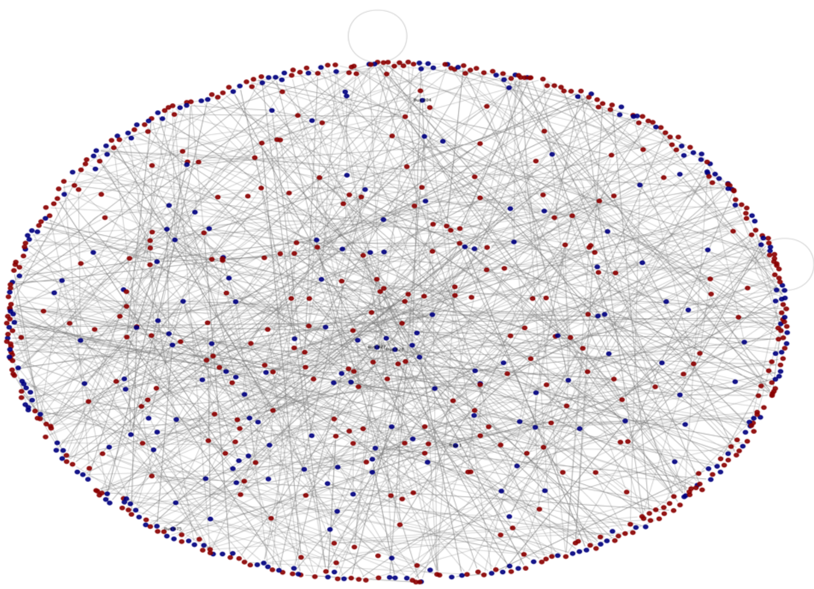

Figure 4.1: Topological Representation of the NGET Transmission Network (Individual Nodes Not Identifiable at this Scale)

While the individual bus nodes cannot be distinguished at this scale, the figure demonstrates that the system is highly interconnected and meshed, consistent with expectations for a national transmission network. Such meshed connectivity is vital for system redundancy and operational flexibility, ensuring that power flows can be re-routed during contingencies.

4.1.3 DCOPF Under Different Injections

In the base network model with 784 buses, the overall bus voltage angle spread is 45.0°, consistent with a meshed transmission system where power flows are shared across parallel paths, limiting large regional separations.

When Scotland and offshore generation (12.5 GW total, including 8.15 GW wind and 1.27 GW nuclear) and interconnectors (9.8 GW total, with 8.8 GW import and 1.0 GW export) are introduced, the spread rises to 69.2°. These concentrated injections and withdrawals alter flow distribution, making the system behave more like a radial network. Instead of dispersing evenly, bulk transfers are channelled through a limited number of north–south corridors, particularly across the Harker–Moffat boundary.

This reflects a known operational feature of the GB system: wind generation from Scotland and the North is often constrained by southward transfer capacity, leading to curtailment during periods of high output. The wider angle spread in the DC OPF results therefore captures both the mathematical effect of concentrated sources/sinks and a key real-world limitation in accommodating northern wind under current transmission capacity.

4.2 Reference Case Studies

To assess the validity of the model, results were benchmarked against Grid Templar reference data [48].

4.2.1 Generation dispatch

System loading was adjusted for seasonal conditions, with interconnector contributions fixed to match the reference dataset. Three cases were examined:

4.2.1.1 Case 1: Winter Base Case (Normal Availability)

Figure 4.2 compares the generation mix from the Grid Templar dataset and the modelled dispatch, both under winter peak loading. Each is normalised to the same system demand of ~36 GW.

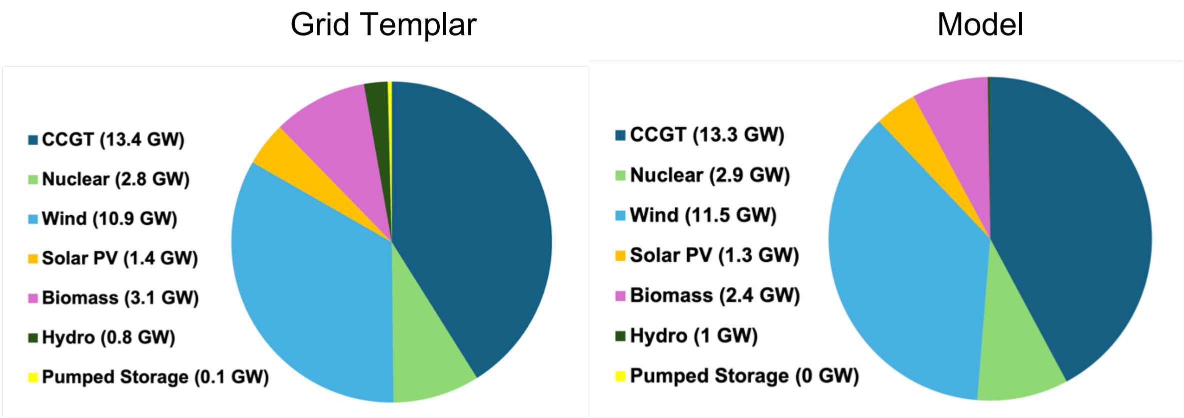

Figure 4.2: Grid Templar vs Model (Normal Availability)

Both the dataset and the model show a generation mix dominated by CCGT (37–38%), wind (30–32%), and nuclear (~8%), with solar at ~4%. Minor deviations occur in smaller categories: biomass (8.6% vs 6.7%) and hydro plus pumped storage (0.2% vs ~2.5%). Despite these, total generation matches demand, and the main technology shares align closely, confirming that the model accurately represents winter operation and that the DC OPF and dataset parsing are functioning correctly.

4.2.1.2 Case 2: Winter Case (No Solar Availability)

Figure 4.3 compares the generation mix from the Grid Templar dataset with the model dispatch under winter loading. In reality, solar is unavailable in this case, and the grid templar dataset records 0 GW from solar. The model, however, still dispatches around 1.3 GW, treating it as fully available. This limitation is not unique to solar; it also applies to wind, which the model dispatches whenever capacity exists, regardless of whether weather conditions allow it.

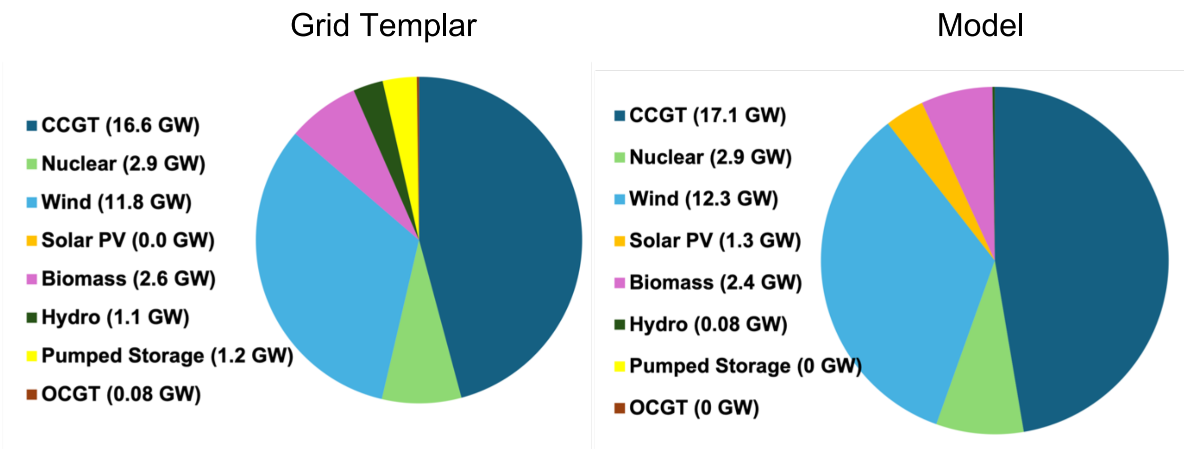

Figure 4.3: Grid Templar vs Model (No Solar Availability)

The overall generation mix is otherwise very similar. CCGT contributes 45.8% versus 47.3% while wind contributes 32.6% percent versus 34%, and nuclear 7.85% versus 8.12%. The close match for the main technologies shows that the model captures broad system behaviour under winter conditions. However, the inclusion of solar when none is available highlights a structural weakness: generation is dispatched by installed capacity rather than availability, overstating renewable penetration and masking the true role of hydro plants.

4.2.1.3 Case 3: Summer Case (High Solar Availability)

Figure 4.4 compares the generation mix from the Grid Templar dataset with the model dispatch under summer conditions. The model dataset reports only 1.3 GW capacity of solar, and this is exactly what the model dispatches. In reality, solar availability in summer is much higher (over 10 GW recorded by Grid Templar on the same day), meaning model underestimates solar output. Similarly, the ETYS appendix dataset reports no pumped storage contribution, which also carries through to the model.

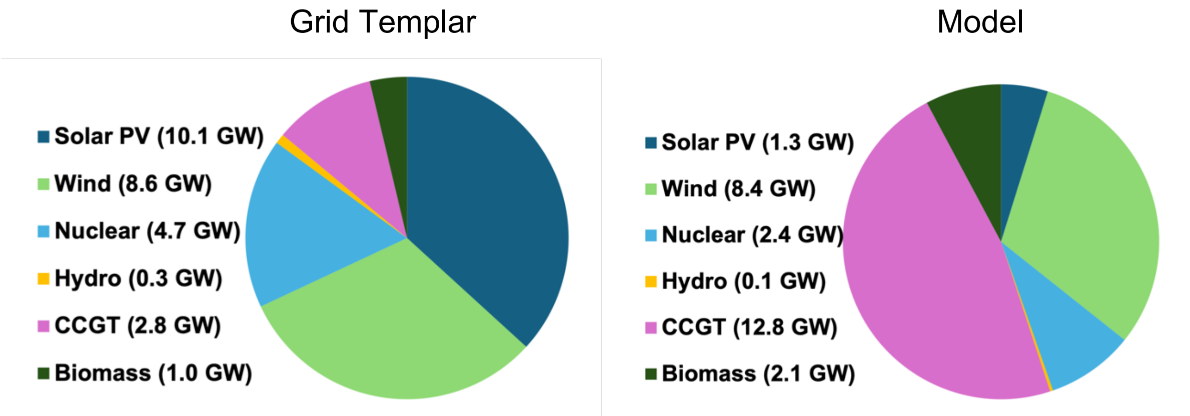

Figure 4.4: Grid Templar vs Model (High Solar Availability)

Other technologies are more consistent. For example, wind is 8.6 GW in the grid templar dataset compared with 8.4 GW in the model. Nuclear differs more noticeably, with 4.7 GW in the dataset versus 2.4 GW in the model, while biomass is slightly overstated by the model **(**1.0 GW vs 2.1 GW**).** CCGT, however, is severely overstated, with 12.8 GW in the model compared to only 2.8 GW in the dataset. This case demonstrates that the accuracy of the model is fundamentally limited by the dataset it is built on. Since the ETYS dataset only reports 1.3 GW of solar, the model cannot reproduce the true scale of solar generation in summer, even though more than 10 GW was actually available. The same applies to pumped storage, which is entirely absent from the dataset and therefore missing from the model output. These omissions result in a distorted generation mix where nuclear and solar are understated and other sources such as CCGT and biomass are overstated to make up the balance. In practice, this highlights that while the model reflects the dataset reliably, it cannot capture real system behaviour when the dataset is incomplete.

4.2.2 Boundary flow

Across all test cases, the Harker–Moffat 400 kV line was observed to operate close to or at its maximum thermal rating. In the winter base case, the line was consistently saturated, with one circuit reaching 100% of its thermal limit and the second circuit operating at 98% loading, as shown in table 4.4. This behaviour is consistent with NESO reports, which state that the north–south B6 boundary capability is limited to 6.7 GW due to thermal constraints on this corridor [49]. The results therefore confirm that the model correctly identifies the Harker–Moffat line as the critical constraint on the B6 boundary, aligning with real system behaviour where north–south transfers are structurally bottlenecked by this corridor.

Table 4.4: Winter Case – Harker–Moffat Line Loading

	from_bus	to_bus	loading(%)
358	SPT	HARK41	100
357	SPT	HARK41	98.05694

In contrast, the summer case presented a different outcome. With 1,400 MW imported from Norway through the Blyth station (North Sea Link), the dominant congestion shifted southwards. The Hartlepool–Tod Point–Lackenby corridor emerged as the critical bottleneck, with loadings of 98% and 90% recorded on its circuits, while Harker–Moffat reduced to around 84–83%. This result highlights how interconnector flows can significantly alter the stress distribution on the NGET network, transferring the binding constraint from the B6 boundary towards the B7 region.

Table 4.5: Summer Case – Key Line Loadings

	from_bus	to_bus	loading(%)
380	HATL21	TODP21	98.3
457	LACK21	TODP21	90.2
358	SPT	HARK41	84.9
357	SPT	HARK41	83.2

The location of this constraint within NGET, between the B6 and B7 boundaries, is illustrated in Figure 4.5, which was adapted from Elexon boundary diagrams. It shows how the Blyth interconnector connects into Hartlepool and flows south through Tod Point to Lackenby, reinforcing the modelled outcome that this corridor becomes critical under high interconnector imports.

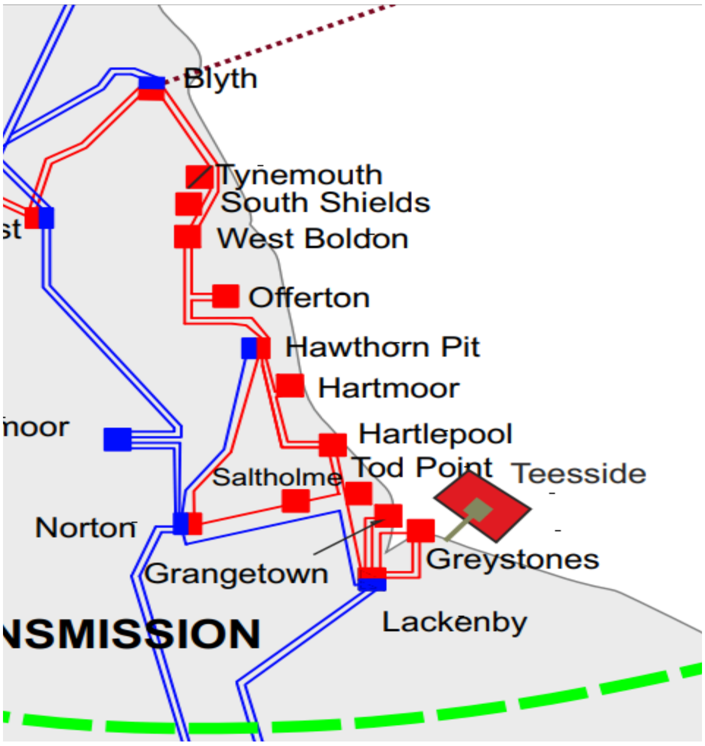

Figure 4.5: B6–B7 Boundary Constraint: Blyth to Hartlepool–Lackenby (adapted from Elexon [50])

Further analysis of this boundary would require access to more detailed operational data, including reinforcement plans, dynamic ratings, and curtailment records, which were not available within the scope of this study. Nevertheless, these findings demonstrate that the model captures key operational constraints observed in practice and provides a reliable basis for subsequent congestion and renewable integration studies.

4.3 Scenario Analysis: Increased Renewables and Loading (+10%)

Wind output rises from 12,250.6 MW to 13,475.6 MW, while solar increases from 1324.5 MW to 1456.9 MW. Other generation categories remain unchanged.

The DCOPF results also show that the overall network angle spread increases from 76 degrees in the base case to 79 degrees in the scenario, a change of about 3 degrees, when compared to the winter base case.

The extra 1.35 GW of renewable generation is absorbed into the system without displacing thermal units, but the wider angle spread indicates greater stress on inter-area transfers. In particular, the Harker–Moffat corridor remains binding at its 6.7 GW thermal limit, confirming it as the critical boundary in both cases.

The 3-degree increase in overall angle separation is modest in absolute terms but important in context. It shows that additional renewable injections amplify north–

south transfer requirements, reinforcing existing congestion and widening the angle difference between exporting and importing areas. In practical terms, this means that even relatively small increases in renewable penetration can exacerbate bottlenecks that are already binding, increasing congestion costs and reducing operational flexibility.

4.4 N-1 Security Assessment

The N-1 assessment for the winter case yields a maximum supported load of 8,754.91 MW, specific to the NGET system (England and Wales).

At first glance, the result is implausibly low, since a winter GB system should normally support several tens of gigawatts of demand. However, when considering that parallel circuits in the dataset are represented as single lines, the outcome becomes more understandable. The aggregation of parallel paths into a single equivalent line significantly reduces apparent transfer capability and therefore constrains the N-1 secure load.

That said, the result may also be influenced by dataset inconsistencies or data errors, such as misapplied thermal ratings or transformer parameters. While the figure of 8.75 GW can be rationalised in terms of the simplified representation of parallel lines, it ultimately does not reflect the true behaviour or secure capacity of the GB transmission system. Instead, it highlights the limitations of the dataset and modelling assumptions used in this study.

4.5 Limitations and Challenges of the Approach

From this chapter, several key points emerge about the model and its application:

- The model dispatches renewables as if they are readily available, rather than reflecting real weather conditions. When the right conditions are set in the dataset, it can be applied to scenario studies that provide credible insights into congestion and dispatch.
- The model is only as good as the dataset it relies on. Incomplete or constrained datasets lead directly to unrealistic outputs.
- Scenario analysis demonstrates that the framework can be used to test the effects of increased load or renewable penetration, but the outcomes remain tied to the quality and realism of the input data.

- Economic representation is limited. Results reflect market behaviour only in the sense of least-cost dispatch, and the use of LCOE-based cost functions captures long-term averages but ignores short-term bidding strategies, ancillary services, and operational constraints.

In summary, the model is useful for analysing active power flows, identifying congestion points, and exploring scenarios, but its validity is constrained by the underlying dataset and by the simplifications of the DC OPF approach.

4.6 Summary

The model was required to meet the following criteria:

- Correctly parse the dataset and generate accurate network elements.
- Assemble a functioning transmission model capable of DCOPF.
- Run analyses and scenarios reliably.
- Produce results that align with reference data.

Since all of these criteria were satisfied, the model can be considered successful.

Nonetheless, some limitations were observed. Renewable technologies were dispatched as if they were always fully available, meaning seasonal and weather-driven effects were not reflected. The accuracy of the results was also tied directly to the completeness of the dataset: where inputs such as solar or pumped storage were under-represented, the model output could not capture real system behaviour.

The objectives of the study were also addressed:

4. Validate the model against reference sources – completed successfully.
5. Apply the model to scenarios, constraint identification, and N-1 studies – demonstrated through selected examples. However, only illustrative scenarios were analysed, and a full Future Energy Scenarios study was beyond the scope of this work.

In summary, this chapter has shown that the model provides a reliable and tractable framework for representing the GB transmission system. It has been validated against trusted sources and shown to reproduce key operational constraints. The framework is therefore well suited for exploring congestion and testing scenarios, though its reliance on dataset completeness and simplified treatment of renewable availability means its outputs must be interpreted with caution.

5 Conclusions and future work

This chapter brings together the main findings of the study, assessing how well the stated objectives were achieved and outlining potential directions for future research and development.

5.1 Conclusions

This dissertation set out to develop and validate an automated model of the GB transmission system, with a focus on the NGET network. The central objectives were to select an appropriate modelling framework, process large public datasets, construct and automate a reproducible model, validate it against reference sources, and apply it to representative scenarios including congestion and N-1 security assessments.

Each of these objectives was addressed. An open-source framework based on Python and pandapower was selected and justified as the most suitable tool for automated DCOPF modelling. The NESO ETYS appendices were processed to create buses, lines, transformers, loads, and generators in a consistent, reproducible manner. The parsing procedures successfully retained technical attributes such as line reactances and transformer ratings, and the automated scripts were able to regenerate the model directly from raw datasets. Validation against Grid Templar reference data and NESO boundary reports confirmed that the model captured key features of real system behaviour, including generation dispatch patterns and congestion on the B6 boundary. In this sense, the framework satisfied the pass–fail criteria set at the outset and can be considered a successful representation of the NGET transmission network.

However, some gaps remain. Validation was limited by the quality and completeness of available datasets: solar and pumped storage were understated, reducing their apparent contribution, and parallel circuits were aggregated into single lines, constraining transfer capability and distorting N-1 results. Validation also relied solely on dispatch data, as no published line flow or related information was available. Renewables were dispatched as if always available, meaning weather-driven availability was not represented. Addressing these limitations would require additional datasets and validation against time-series data rather than static snapshots.

The study also aimed to apply the model to scenarios, constraint identification, and N-1 assessments. These were demonstrated successfully: congestion points were identified at the line level, the impact of additional renewable penetration was explored, and N-1 limits were tested. At the same time, it must be acknowledged that only selected scenarios were analysed to illustrate the model's potential. A full exploration of the Future Energy Scenarios was not carried out, but the work has shown that the framework is capable of such studies when extended further.

Taken together, the work has achieved its core objectives. It has demonstrated that an automated, reproducible transmission model can be built directly from public datasets and validated against trusted sources. While its fidelity is bounded by simplifications and data completeness, the framework establishes a sound foundation for future extensions and applications.

5.2 Future work

Looking ahead, there are several extensions that could enhance both the fidelity and applicability of this modelling framework. These can be approached in increasing levels of ambition, and each builds naturally on the foundations established here:

Automated preprocessing of generator data using AI methods
Generator allocation was a significant bottleneck in this study, relying on fuzzy matching between project names and substation identifiers. An extension would be to train or apply large language models (LLMs) to pre-process generator connection data, automatically matching ambiguous entries to their correct substations. This aligns with emerging work in applying machine learning for dataset cleaning and asset mapping in energy systems. For this project, such a method would improve the reproducibility and accuracy of generator placement, producing cleaner, more reliable inputs for OPF studies.

Scenario expansion and probabilistic analysis
The present study demonstrated the framework using selected scenarios but did not undertake a full Future Energy Scenarios (FES) exploration. By automating scenario scaling and embedding probabilistic sampling of demand and renewable conditions, this framework could be extended to run hundreds of cases efficiently. The expected output would be not just single-point studies, but distributions of outcomes, allowing insight into the likelihood of congestion events, boundary violations, or N-1 insecurities.

Higher-resolution inclusion of Scotland and offshore systems

In this study, Scotland and offshore wind were represented as aggregated injections. Extending the model to include the SP Transmission and SHE Transmission networks at bus-level, and modelling offshore transmission nodes explicitly, would bring the representation closer to the operational models maintained by NGESO. The expected output would be a unified GB-wide model capable of resolving north–south transfer constraints and offshore integration challenges more precisely.

Extension to AC OPF and voltage/stability analysis

The current DCOPF approach prioritises scalability and speed but omits reactive power and voltage behaviour. Moving towards AC OPF would enable inclusion of voltage magnitudes, losses, and reactive flows. This would significantly increase computational complexity, but would extend the framework's scope to cover voltage security, reactive margins, and stability constraints. The expected output would be a model applicable not just to active power flows and congestion studies, but also to operational planning and reinforcement assessments.

References

[1] “National Electricity Transmission System Performance Report”.

[2] “Britain’s reliance on coal-fired power set to end after 140 years.” Accessed: Aug. 20, 2025. [Online]. Available: https://www.ft.com/content/5164185d-b0d6-40d1-99b4-59f8039111c2

[3] Molly Lempriere and Simon Evans, “Q&A: How the UK became the first G7 country to phase out coal power,” CarbonBrief. Accessed: Aug. 20, 2025. [Online]. Available: https://interactive.carbonbrief.org/coal-phaseout-UK/index.html

[4] “Flexibility in the GB Power System,” 2025.

[5] “£58bn plan to rewire Great Britain expected to spark tensions along route | Energy industry | The Guardian.” Accessed: Aug. 20, 2025. [Online]. Available: https://www.theguardian.com/business/2024/mar/19/58bn-plan-rewire-great-britain-spark-tensions-route

[6] “UK regulator fast tracks $5.2 billion energy grid investment | Reuters.” Accessed: Aug. 20, 2025. [Online]. Available: https://www.reuters.com/business/energy/uk-regulator-fast-tracks-52-billion-energy-grid-investment-2025-03-20/

[7] “Future Energy Scenarios: Pathways to Net Zero.” Accessed: Aug. 22, 2025. [Online]. Available: https://www.neso.energy/document/364541/download

[8] A. Lyden *et al.*, “PyPSA-GB: An open-source model of Great Britain’s power system for simulating future energy scenarios,” *Energy Strategy Reviews*, vol. 53, p. 101375, May 2024, doi: 10.1016/j.esr.2024.101375.

[9] L. Franken *et al.*, “Power system benefits of simultaneous domestic transport and heating demand flexibility in Great Britain’s energy transition,” 2024, doi: 10.1016/j.apenergy.2024.124522.

[10] A. F. Lyden *et al.*, “An Open-Source Model of Great Britain’s Power System for Simulating Future Energy Scenarios,” 2023, doi: 10.2139/SSRN.4509311.

[11] A. B. Birchfield *et al.*, “A Metric-Based Validation Process to Assess the Realism of Synthetic Power Grids,” *Energies 2017, Vol. 10, Page 1233*, vol. 10, no. 8, p. 1233, Aug. 2017, doi: 10.3390/EN10081233.

[12] L. Franken *et al.*, “Power system benefits of simultaneous domestic transport and heating demand flexibility in Great Britain’s energy transition,” *Appl Energy*, vol. 377, p. 124522, Jan. 2025, doi: 10.1016/J.APENERGY.2024.124522.

[13] “Electricity Ten Year Statement,” 2024, Accessed: Aug. 21, 2025. [Online]. Available: https://www.neso.energy/document/352001/download

[14] Chris Owens, “National Grid’s Summer Outlook 2023 | the Blackout report.” Accessed: Aug. 21, 2025. [Online]. Available: https://www.theblackoutreport.co.uk/2023/04/20/national-grid-summer-outlook-2023/?

[15] “Grid constraints cost UK £1 billion per year in wasted wind power generation | Windpower Monthly.” Accessed: Aug. 29, 2025. [Online]. Available: https://www.windpowermonthly.com/article/1860633/grid-constraints-cost-uk-1-billion-per-year-wasted-wind-power-generation?

[16] “2025 Annual Balancing Costs Report,” 2025.

[17] “Britain’s Electricity Explained: 2023 Review | National Energy System Operator.” Accessed: Aug. 21, 2025. [Online]. Available: https://www.neso.energy/news/britains-electricity-explained-2023-review?

[18] J. J. Grainger and W. D. Stevenson, “POWER SYSTEM ANALYSIS,” 1994.

[19] B. Stott and O. Alsac, “Fast decoupled load flow,” *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-93, no. 3, pp. 859–869, 1974, doi: 10.1109/TPAS.1974.293985.

[20] B. Stott, J. Jardim, and O. Alsaç, “DC power flow revisited,” *IEEE Transactions on Power Systems*, vol. 24, no. 3, pp. 1290–1300, 2009, doi: 10.1109/TPWRS.2009.2021235.

[21] Allen J. Wood, Bruce F. Wollenberg, and Gerald B. Sheblé, “Power Generation, Operation, and Control, 3rd Edition | Wiley.” Accessed: Aug. 21, 2025. [Online]. Available: https://www.wiley.com/en-us/Power+Generation%2C+Operation%2C+and+Control%2C+3rd+Edition-p-9780471790556

[22] H. W. Dommel and W. F. Tinney, “Optimal Power Flow Solutions,” *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-87, no. 10, pp. 1866–1876, 1968, doi: 10.1109/TPAS.1968.292150.

[23] S. Frank, I. Steponavice, and S. Rebennack, “Optimal power flow: a bibliographic survey I,” *Energy Systems 2012 3:3*, vol. 3, no. 3, pp. 221–258, Apr. 2012, doi: 10.1007/S12667-012-0056-Y.

[24] D. K. Molzahn and I. A. Hiskens, “A Survey of Relaxations and Approximations of the Power Flow Equations,” *Foundations and Trends® in Electric Energy Systems*, vol. 4, no. 1–2, pp. 1–221, Feb. 2019, doi: 10.1561/3100000012.

[25] “PSS®E – transmission planning and analysis.” Accessed: Aug. 21, 2025. [Online]. Available: https://www.siemens.com/global/en/products/energy/grid-software/planning/pss-software/pss-e.html

[26] “PowerFactory - DIgSILENT.” Accessed: Aug. 21, 2025. [Online]. Available: https://www.digsilent.de/en/powerfactory.html

[27] “Overview | PSCAD.” Accessed: Aug. 21, 2025. [Online]. Available: https://www.pscad.com/software/pscad/overview

[28] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, “MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education,” *IEEE Transactions on Power Systems*, vol. 26, no. 1, pp. 12–19, Feb. 2011, doi: 10.1109/TPWRS.2010.2051168.

[29] L. Thurner *et al.*, “Pandapower - An Open-Source Python Tool for Convenient Modeling, Analysis, and Optimization of Electric Power Systems,” *IEEE Transactions on Power Systems*, vol. 33, no. 6, pp. 6510–6521, Nov. 2018, doi: 10.1109/TPWRS.2018.2829021.

[30] T. Brown, J. Hörsch, and D. Schlachtberger, “PyPSA: Python for power system analysis,” *J Open Res Softw*, vol. 6, no. 1, 2018, doi: 10.5334/jors.188.

[31] L. P. Kunjumuhammed, B. C. Pal, and N. F. Thornhill, “A Test System Model for Stability Studies of UK Power Grid”.

[32] J. F. DeCarolis, K. Hunter, and S. Sreepathi, “The case for repeatable analysis with energy economy optimization models,” *Energy Econ*, vol. 34, no. 6, pp. 1845–1853, Nov. 2012, doi: 10.1016/j.eneco.2012.07.004.

[33] A. M. Foley, B. P. Ó Gallachóir, J. Hur, R. Baldick, and E. J. McKeogh, “A strategic review of electricity systems models,” *Energy*, vol. 35, no. 12, pp. 4522–4530, Dec. 2010, doi: 10.1016/J.ENERGY.2010.03.057.

[34] J. F. DeCarolis, K. Hunter, and S. Sreepathi, “The case for repeatable analysis with energy economy optimization models,” *Energy Econ*, vol. 34, no. 6, pp. 1845–1853, Nov. 2012, doi: 10.1016/J.ENECO.2012.07.004.

[35] “Energy Exemplar to help National Grid ESO shape energy transition.” Accessed: Aug. 29, 2025. [Online]. Available: https://www.energyexemplar.com/blog/national-grid-eso-selects-plexos?

[36] H. Li *et al.*, “Building Highly Detailed Synthetic Electric Grid Data Sets for Combined Transmission and Distribution Systems,” *IEEE Open Access Journal of Power and Energy*, vol. 7, pp. 478–488, 2020, doi: 10.1109/OAJPE.2020.3029278.

[37] A. B. Birchfield, T. Xu, and T. J. Overbye, “Power flow convergence and reactive power planning in the creation of large synthetic grids,” *IEEE Transactions on Power Systems*, vol. 33, no. 6, pp. 6667–6674, Nov. 2018, doi: 10.1109/TPWRS.2018.2813525.

[38] A. B. Birchfield *et al.*, “A Metric-Based Validation Process to Assess the Realism of Synthetic Power Grids,” *Energies 2017, Vol. 10, Page 1233*, vol. 10, no. 8, p. 1233, Aug. 2017, doi: 10.3390/EN10081233.

[39] A. B. Birchfield, T. Xu, K. S. Shetye, and T. J. Overbye, “Building Synthetic Power Transmission Networks of Many Voltage Levels, Spanning Multiple Areas,” *Proceedings of the Annual Hawaii International Conference on System Sciences*, vol. 2018-January, pp. 2766–2774, Jan. 2018, doi: 10.24251/HICSS.2018.349.

[40] K. M. Gegner, A. B. Birchfield, T. Xu, K. S. Shetye, and T. J. Overbye, “A methodology for the creation of geographically realistic synthetic power flow models,” *2016 IEEE Power and Energy Conference at Illinois, PECI 2016*, Apr. 2016, doi: 10.1109/PECI.2016.7459256.

[41] T. Xu, A. B. Birchfield, K. M. Gegner, K. S. Shetye, and T. J. Overbye, “Application of Large-Scale Synthetic Power System Models for Energy Economic Studies,” *Proceedings of the Annual Hawaii International*

Conference on System Sciences, vol. 2017-January, pp. 3123–3129, Jan. 2017, doi: 10.24251/HICSS.2017.386.

[42] A. B. Birchfield, “The creation of synthetic power grids: preliminary considerations,” Nov. 2016, Accessed: Aug. 25, 2025. [Online]. Available: https://hdl.handle.net/2142/95328

[43] A. B. Birchfield, “Inertia Adequacy in Transient Stability Models for Synthetic Electric Grids,” Jul. 2022, Accessed: Aug. 25, 2025. [Online]. Available: https://arxiv.org/pdf/2207.03396

[44] H. Li *et al.*, “Building Highly Detailed Synthetic Electric Grid Data Sets for Combined Transmission and Distribution Systems,” *IEEE Open Access Journal of Power and Energy*, vol. 7, pp. 478–488, 2020, doi: 10.1109/OAJPE.2020.3029278.

[45] “Interconnector Register - 26 August 2025 | National Energy System Operator.” Accessed: Aug. 26, 2025. [Online]. Available: https://www.neso.energy/data-portal/interconnector-register/interconnector_register_-_26_august_2025

[46] E. Press *et al.*, “Renewable Power Generation Costs in 2020,” 2021, Accessed: Aug. 23, 2025. [Online]. Available: www.irena.org

[47] “Projected Costs of Generating Electricity,” 2015, Accessed: Aug. 23, 2025. [Online]. Available: www.iea.org/t&c/

[48] “U.K. National Grid status download data.” Accessed: Aug. 28, 2025. [Online]. Available: https://www.gridwatch.templar.co.uk/download.php

[49] “Scottish boundaries | National Energy System Operator.” Accessed: Aug. 28, 2025. [Online]. Available: https://www.neso.energy/publications/electricity-ten-year-statement-etys/electricity-transmission-network-requirements/scottish-boundaries

[50] “Elexon GB transmission system boundaries.” Accessed: Aug. 28, 2025. [Online]. Available: https://www.elexon.co.uk/bsc/documents/data/operational-data/gb-transmission-system-boundary-zone-map/

Appendicies

Appendix A - Project outline

Introduction

This document provides a project outline for a dissertation focused on modelling the future of GB power using a script-based approach. It outlines the motivation, scope, aims, objectives, and a brief project plan. The goal is to automate the creation of a validated network model from public datasets, with potential applications in future energy scenario analysis.

Motivation

The UK electricity network is undergoing a significant transformation due to increasing integration of low-carbon technologies, decentralised generation, and electrification of demand, particularly through electric vehicles (EVs) and heat pumps. As a result, transmission network modelling is becoming increasingly complex and essential for maintaining grid stability and planning future infrastructure investments. Traditional modelling methods are time-consuming and manually intensive, making it challenging to iterate and evaluate multiple long-term energy scenarios efficiently.

Automating the modelling of the Great Britain (GB) transmission network using publicly available datasets can enable rapid scenario analysis, support future energy planning, and contribute to better understanding of the operational and investment implications under varying assumptions. This dissertation contributes to this area by developing a reusable and extensible modelling script, facilitating integration with power system analysis tools and future scenario projections.

Project Scope

The project focuses on the development of a script, written in a suitable language such as Python or MATLAB, capable of:

- Extracting and processing GB transmission network data from publicly available sources (e.g., NESO appendices, ETYS).
- Creating a network model that captures key electrical characteristics (such as busbars, generators, transformers, interconnectors, overhead lines, underground cables, and loads).
- Validating the generated network model against existing system benchmarks or simplified reference models.

- Performing DC Optimal Power Flow (DC OPF) analysis using the generated model to assess power flows under defined operating conditions.
- Providing a flexible framework that can be extended to simulate Future Energy Scenarios (FES) by National Grid ESO or user-defined scenarios.

Exclusions from Scope:

- Real-time data modelling.
- Full integration with operational SCADA or Energy Management Systems.

Aim

To develop a DC Optimal Power Flow (DC-OPF) model of the Great Britain (GB) transmission network using a suitable programming language and publicly available power system data.

Objectives

1. To extract and process relevant network data (branch, generation data, etc)
2. To develop a representative model of the GB transmission network.
3. To implement the DC-OPF model using a suitable programmable environment.
4. To validate the model's performance by comparing results with known system characteristics or published data.
5. To explore the impact of changes in system parameters or generation patterns on optimal power flow and network reliability.
6. To provide documentation to support reuse, reproducibility, and future work.

Methodology Overview:

1. Collect and process publicly available data from NESO's Electricity Ten Year Statement (ETYS), including bus, branch, and generator information.
2. Construct a simplified DC model of the GB transmission network using Python or MATLAB.
3. Formulate and solve the DC Optimal Power Flow (DC-OPF) to determine optimal generator dispatch under network constraints.
4. Validate the model by comparing its output with known system characteristics or published benchmarks.
5. Perform scenario analysis to assess the impact of changes in network parameters or generation patterns on power flow and system reliability.

Brief Project Plan

- **Week 1–2: Literature Review**
  Review relevant studies on DC-OPF, GB power system structure, and NESO data sources.
- **Week 3–4: Data Collection and Processing**
  Extract and clean transmission network data from NESO ETYS.

- **Week 5–6: Model Construction**
  Build a simplified DC representation of the GB network in Python.
- **Week 7: OPF Implementation**
  Formulate and solve the DC-OPF using an appropriate solver or optimization library.
- **Week 8: Model Validation**
  Compare model results with known benchmarks or system characteristics.
- **Week 9: Scenario Analysis**
  Simulate the impact of different generation patterns or network changes.
- **Week 10–12: Writing and Finalization**
  Compile findings, complete the dissertation report, and conduct proofreading.

Expected Deliverables

- A well-documented script for GB network modelling.
- A validated power system model of the GB transmission network.
- Scenario simulation results and discussion.
- A dissertation summarising methodology, outcomes, limitations, and recommendations for future work.

Appendix B - Risk assessment

Risk Assessment Statement for Dissertation Project (Coding-Based Work)

This risk assessment covers general office-related risks associated with the completion of my dissertation, which involves primarily computer-based tasks such as programming, data analysis, and documentation.

Nature of Work:
The project involves prolonged use of a computer for coding, data processing, report writing, and documentation, carried out in a standard office or home office setting.

Identified Risks and Control Measures:

Hazard	**Risk**	**Persons at Risk**	**Control Measures**
Prolonged computer use	Eye strain, repetitive strain injury (RSI), fatigue	Self	• Follow 20-20-20 rule. • Take regular breaks. • Adjust screen brightness. • Use ergonomic setup.
Poor posture or seating	Back, neck, or shoulder pain	Self	• Use an adjustable chair. • Maintain good posture. • Ensure screen is at eye level.
Electrical equipment	Shock or fire risk	Self	• Ensure all equipment is PAT tested (if required). • Do not overload sockets. • Keep liquids away from electronics.
Slips, trips, and falls	Physical injury	Self and others in the room	• Keep cables tidy and out of walkways. • Ensure floor area is clear
Mental stress or overwork	Fatigue, reduced productivity	Self	• Maintain a work-life balance. • Set realistic goals and schedules. • Seek support if needed.

Conclusion:
With the outlined control measures in place, the risks associated with this project are minimal and manageable. The working environment will be kept safe through routine awareness and adherence to standard office health and safety practices.

Appendix C - Python Implementation (Pandapower Framework)

C.1 network.py

```
import pandapower as pp
import warnings

from user_input import network_file, demand_file,
generator_file, bus_sheet, substation_sheet,
transformer_sheet, line_sheet, load_sheet, gen_sheet

from elements.buses import create_buses
from elements.lines import create_lines
from elements.transformers import create_transformers
from elements.loads import create_loads,
group_bus_by_substation
from elements.generators import create_gens

from validation_elements.interconnectors import
create_interconnectors
from validation_elements.spt import create_spt_assets

from utilities.run_dc import run_dcopf
from utilities.plotting import plot_network

from tests.security import n1_security
from tests.generation_summary import generation_summary
from tests.get_results import get_results


warnings.filterwarnings('ignore')

# --- Create Empty Network ---
net = pp.create_empty_network()
```

```
# --- Create Elements ---

SHE_BUS, SPT_BUS, OFTO_BUS, NGET_bus_lookup =
create_buses(net, network_file, bus_sheet)
create_lines(net, NGET_bus_lookup, SHE_BUS, SPT_BUS, OFTO_BUS,
network_file, line_sheet)
create_transformers(net, NGET_bus_lookup, SHE_BUS, SPT_BUS,
OFTO_BUS, network_file, transformer_sheet)
substation_group = group_bus_by_substation(NGET_bus_lookup)
total_SHE_load, total_SPT_load = create_loads(net,
NGET_bus_lookup, substation_group, demand_file, load_sheet)
create_gens(net, NGET_bus_lookup, substation_group,
generator_file, gen_sheet, network_file, substation_sheet)

# For validation
create_interconnectors(net, NGET_bus_lookup, mode="import")
create_spt_assets(net, SPT_BUS, total_SHE_load,
total_SPT_load)

EXT_GRID = pp.create_ext_grid(net, SPT_BUS, vm_pu=1, name="SPT
BUS")
net.ext_grid.at[EXT_GRID, "max_p_mw"] = 0  # max import
net.ext_grid.at[EXT_GRID, "min_p_mw"] = 0

# Apply load scaling
net.load.loc[:, 'p_mw'] *= 1.1

# Apply renewables scaling
net.gen.loc[net.gen["name"].str.contains("Wind|Solar PV"),
"p_mw"] *= 1.1

# Run DCOPF and get network results with DCOPF results
run_dcopf(net)
get_results(net)
```

```
# # Generation summary
generation_summary(net)

# N-1 Security on network
secure_scale, violations = n1_security(net, step=0.01,
max_scale=2.0, include_impedances=True)

print(f"Maximum secure load multiplier: {secure_scale:.2f}x
base")
if not violations.empty:
    print("Violations at failure step:")
    print(violations.sort_values("max_loading_percent",
ascending=False).head(10))

# Network topology plot
plot_network(net)
```

C.2 buses.py

```
import pandapower as pp
import pandas as pd
from elements.substations import NGET_SUBSTATIONS



# === Bus Creation Function ===

def create_buses(net, network_file, bus_sheet):
    def NGET_buses():

        bus_names = []
        sheet_names = bus_sheet

        for sheet_name in sheet_names:
            df = pd.read_excel(network_file,
sheet_name=sheet_name, skiprows=1)

            for idx in df.index:
                bus_names.append(df.at[idx, "Node 1"])
                bus_names.append(df.at[idx, "Node 2"])

        buses = list(dict.fromkeys(bus_names))
        return buses

    # Creating (slack) Buses for other sub networks
    SHE_BUS = pp.create_bus(net, vn_kv=400, name="SHE_BUS")
    SPT_BUS = pp.create_bus(net, vn_kv=400, name="SPT_BUS")
    OFTO_BUS = pp.create_bus(net, vn_kv=400, name="OFTO_BUS")

    buses = NGET_buses()
    NGET_bus_lookup = {}

    # create bus
```

```
    for bus in buses:
        if bus[:4] in NGET_SUBSTATIONS:
            bus_idx = pp.create_bus(net, vn_kv=400, name=bus)
            NGET_bus_lookup[bus] = bus_idx # Store in lookup
table

    print("Bus creation complete.")
    return SHE_BUS, SPT_BUS, OFTO_BUS, NGET_bus_lookup
```

C.3 substations.py

```
import pandas as pd
from user_input import network_file, substation_sheet

# Initialize empty sets for each operator's substations (sets
prevent duplicates)
NGET_SUBSTATIONS = set()
SHE_SUBSTATIONS = set()
SPT_SUBSTATIONS = set()
OFTO_SUBSTATIONS = set()

# Define each operator with their respective sheet name and
substation set reference
operators = [
    {
        "name": "NGET",
        "sheet_name": substation_sheet[2],
        "substations": NGET_SUBSTATIONS
    },
    {
        "name": "SHE",
        "sheet_name": substation_sheet[0],
        "substations": SHE_SUBSTATIONS
    },
    {
        "name": "SPT",
        "sheet_name": substation_sheet[1],
        "substations": SPT_SUBSTATIONS
    },
    {
        "name": "OFTO",
        "sheet_name": substation_sheet[3],
        "substations": OFTO_SUBSTATIONS
    },
]
```

```
# Read substation data from each operator's sheet and populate
the corresponding set
for operator in operators:
    df = pd.read_excel(network_file,
sheet_name=operator["sheet_name"], skiprows=1)
    for idx in df.index:
        site_code = df.at[idx, "Site Code"]
        operator["substations"].add(site_code)
```

C.4 transformers.py

```
import pandapower as pp
import pandas as pd
from elements.substations import NGET_SUBSTATIONS,
SHE_SUBSTATIONS, SPT_SUBSTATIONS, OFTO_SUBSTATIONS

# === Transformer Creation Function ===

def create_transformers(net, NGET_bus_lookup, SHE_BUS,
SPT_BUS, OFTO_BUS, network_file, transformer_sheet):
    df = pd.read_excel(network_file,
sheet_name=transformer_sheet, skiprows=1)
    for idx in df.index:
        from_bus_name = df.at[idx, "Node 1"]
        to_bus_name = df.at[idx, "Node 2"]
        x_pu = df.at[idx, "X (" + '%' + " on 100MVA)"]/100
        mva_rating = df.at[idx, "Rating (MVA)"]

        # Assign bus indices
        if from_bus_name[:4] in NGET_SUBSTATIONS:
            from_bus = NGET_bus_lookup[from_bus_name]
        elif from_bus_name[:4] in SHE_SUBSTATIONS:
            from_bus = SHE_BUS
        elif from_bus_name[:4] in SPT_SUBSTATIONS:
            from_bus = SPT_BUS
        elif from_bus_name[:4] in OFTO_SUBSTATIONS:
            from_bus = OFTO_BUS
        else:
            print("Unhandled from_bus:", from_bus_name)
            continue

        if to_bus_name[:4] in NGET_SUBSTATIONS:
            to_bus = NGET_bus_lookup[to_bus_name]
        elif to_bus_name[:4] in SHE_SUBSTATIONS:
            to_bus = SHE_BUS
```

```
        elif to_bus_name[:4] in SPT_SUBSTATIONS:
            to_bus = SPT_BUS
        elif to_bus_name[:4] in OFTO_SUBSTATIONS:
            to_bus = OFTO_BUS
        else:
            print("Unhandled to_bus:", to_bus_name)
            continue

        # Create transformer in the form of an impedance
element
        pp.create_impedance(net,
            from_bus=from_bus,
            to_bus=to_bus,
            rft_pu=0,
            xft_pu=x_pu,
            sn_mva=mva_rating
        )
    print("Transformer creation complete.")
```

C.5 loads.py

```
import pandapower as pp
import pandas as pd
from collections import defaultdict
from elements.substations import NGET_SUBSTATIONS,
SHE_SUBSTATIONS, SPT_SUBSTATIONS, OFTO_SUBSTATIONS

# === Utility Functions ===

def group_bus_by_substation(NGET_bus_lookup):
    substation_group = defaultdict(list)

    for bus_name, bus_idx in NGET_bus_lookup.items():
        substation_name = bus_name[:4]
        if substation_name in NGET_SUBSTATIONS:
            substation_group[substation_name].append(bus_name)
    substation_group = dict(substation_group)
    return substation_group

# === Load Creation Function ===

def create_loads(net, NGET_bus_lookup, substation_group,
demand_file, load_sheet):

    # === Initialize accumulators and containers ===

    load_per_substation = {substation: 0 for substation in
NGET_SUBSTATIONS}
    NGET_LOAD_BUS_NOT_EXISTING = set()



    # extras
    SHE_LOAD_BUS = {}
    SPT_LOAD_BUS = {}
    OFTO_LOAD_BUS = {}
```

```
    NONEXISTENT_LOAD_BUS = {}
    total_NGET_load_not_connected = 0
    total_SHE_load = 0
    total_SPT_load = 0
    total_OFTO_load = 0


    df = pd.read_excel(demand_file, sheet_name=load_sheet,
skiprows=9)
    for idx in df.index:
        bus = df.at[idx, "Node"]
        substation = bus[:4]
        p_mw = df.at[idx, "24/25 MW"]

        if substation in NGET_SUBSTATIONS:
            if bus in NGET_bus_lookup:
                pp.create_load(net, NGET_bus_lookup[bus],
p_mw, controllable=False)
            elif substation in load_per_substation:

                load_per_substation[substation] += p_mw

            else:
                NGET_LOAD_BUS_NOT_EXISTING.add(bus)

                # extras
                total_NGET_load_not_connected += p_mw

        elif bus[:4] in SHE_SUBSTATIONS:
            SHE_LOAD_BUS[bus] = p_mw
            total_SHE_load += p_mw

        elif bus[:4] in SPT_SUBSTATIONS:
            SPT_LOAD_BUS[bus] = p_mw
            total_SPT_load += p_mw
```

```
        elif bus[:4] in OFTO_SUBSTATIONS:
            OFTO_LOAD_BUS[bus] = p_mw
            total_OFTO_load += p_mw
        else:
            NONEXISTENT_LOAD_BUS[bus] = p_mw
            total_load_missing += p_mw

    # Distirbuting load evenly among buses

    for substation, total_load in load_per_substation.items():
        buses = substation_group.get(substation, [])

        if not buses:
            # print("Buses not found for substation",
substation)
            continue

        load_per_bus = total_load/len(buses)

        for bus_name in buses:
            bus_idx = NGET_bus_lookup[bus_name]
            if bus_idx is not None:
                pp.create_load(net, bus_idx,
p_mw=load_per_bus, controllable=False)
            else:
                # print(f"Bus {bus_name} not found in
NGET_bus_lookup")
                continue

    print("Load creation complete.")
    return total_SHE_load, total_SPT_load
```

C.6 lines.py

```
import pandapower as pp
import pandas as pd
import math
from elements.substations import NGET_SUBSTATIONS,
SHE_SUBSTATIONS, SPT_SUBSTATIONS, OFTO_SUBSTATIONS


# === Utility Functions ===

def convert_x_from_pu_to_ohm_per_km(x_pu, length_km):
    s_base = 100_000_000  # 100 MVA
    v_base = 400_000      # 400 kV
    z_base = (v_base**2) / s_base
    x_ohm = (x_pu/100) * z_base
    return x_ohm / length_km

def convert_mva_to_ka(mva_rating):
    return ((mva_rating * 1_000_000) / ( math.sqrt(3) *
400_000)) / 1000

# === Line Creation Function ===

def create_lines(net, NGET_bus_lookup, SHE_BUS, SPT_BUS,
OFTO_BUS, network_file, line_sheet):
    df = pd.read_excel(network_file, sheet_name=line_sheet,
skiprows=1)

    for idx in df.index:
        from_bus_name = df.at[idx, "Node 1"]
        to_bus_name = df.at[idx, "Node 2"]
        x_pu = df.at[idx, "X (" + '%' + " on 100 MVA)"]
        ka_rating = convert_mva_to_ka(df.at[idx, "Winter
Rating (MVA)"])
        x_ohm_per_km=0
```

```
        circuit_type = df.at[idx, "Circuit Type"]

       # Determine line length based on circuit type
        if circuit_type in ["OHL", "parallel OHL"]:
            length_km = df.at[idx, "OHL Length (km)"]
        elif circuit_type == "Cable":
            length_km = df.at[idx, "Cable Length (km)"]
            if length_km == 0:
                length_km = 0.01
        elif circuit_type in ["Zero Length", "Series Reactor",
"Series Capacitor", "SSSC"]:
            length_km = 0.01
        elif circuit_type in ["Composite", "parallel
Composite"]:
            length_km = df.at[idx, "OHL Length (km)"] +
df.at[idx, "Cable Length (km)"]
        else:
            print("Unhandled circuit type:", circuit_type)
            continue

        # Calculate reactance
        if x_pu == 0 or length_km == 0:
            x_ohm_per_km = 0.01  # prevent zero reactance
        else:
            x_ohm_per_km =
convert_x_from_pu_to_ohm_per_km(x_pu, length_km)


        # Assign bus indices
        if from_bus_name[:4] in SPT_SUBSTATIONS:
            from_bus = SPT_BUS
            # ka_rating = 1e10  # remove line rating
        elif from_bus_name[:4] in NGET_SUBSTATIONS:
            from_bus = NGET_bus_lookup[from_bus_name]
```

```
elif from_bus_name[:4] in SHE_SUBSTATIONS:
    from_bus = SHE_BUS
elif from_bus_name[:4] in OFTO_SUBSTATIONS:
    from_bus = OFTO_BUS
else:
    print("Unhandled from_bus:", from_bus_name)
    continue


if to_bus_name[:4] in SPT_SUBSTATIONS:
    to_bus = SPT_BUS
elif to_bus_name[:4] in NGET_SUBSTATIONS:
    to_bus = NGET_bus_lookup[to_bus_name]
    # ka_rating = 1e10  # remove line rating
elif to_bus_name[:4] in SHE_SUBSTATIONS:
    to_bus = SHE_BUS
elif to_bus_name[:4] in OFTO_SUBSTATIONS:
    to_bus = OFTO_BUS
else:
    print("Unhandled to_bus:", to_bus_name)
    continue


# Create line
pp.create_line_from_parameters(
    net,
    from_bus=from_bus,
    to_bus=to_bus,
    length_km=length_km,
    r_ohm_per_km=0.0,
    x_ohm_per_km=x_ohm_per_km,
    c_nf_per_km=0.0,
    max_i_ka=ka_rating,
    max_loading_percent=100
```

```
    )
print("Line creation complete.")
```

C.7 generators.py

```
import pandapower as pp
import pandas as pd
from rapidfuzz import process, fuzz
import re



# -----------------
# Helper Functions
# -----------------
def clean_name(name):
    if not isinstance(name, str):
        return ""
    blacklist = [
        "substation", "offshore", "onshore", "station",
"grid",
        "400kv", "275kv", "132kv", "132/33kv",
        "north", "south", "east",
        "wind", "farm", "hydro", "solar"
    ]
    name = name.lower()
    for word in blacklist:
        name = name.replace(word, "")
    name = re.sub(r"[^a-z0-9\s]", "", name)
    name = re.sub(r"\s+", " ", name).strip()
    return name.upper()

def map_to_lcoe_category(plant_type_str):
    if not isinstance(plant_type_str, str):
        return "Other"
    pt = plant_type_str.lower()

    if "coal" in pt:
        return "Coal"
    elif "wind" in pt:
```

```
        return "Wind"
    elif "pv array" in pt or "solar" in pt:
        return "Solar PV"
    elif "nuclear" in pt:
        return "Nuclear"
    elif "hydro" in pt:
        return "Hydro"
    elif "pump storage" in pt or "pumped storage" in pt:
        return "Pumped Storage"
    elif "ccgt" in pt:
        return "CCGT"
    elif "ocgt" in pt:
        return "OCGT"
    elif "chp" in pt:
        return "CHP"
    elif "oil" in pt:
        return "Oil"
    elif "biomass" in pt or "thermal" in pt:
        return "Biomass"
    elif "energy storage" in pt or "battery storage" in pt or
"storage" in pt:
        return "Battery Storage"
    else:
        return "Other"

def get_best_match(name, site_names_clean):
    match = process.extractOne(name, site_names_clean,
scorer=fuzz.WRatio)
    if match:
        return match[0], match[1]
    return None, 0


# -----------------
# Main Generator Creation
```

```
# -----------------
def create_gens(net, NGET_bus_lookup, substation_group,
gen_file, gen_sheet, network_file, substation_sheet):

    def load_substations():
        substations = []
        for sheet in substation_sheet:
            sub_df = pd.read_excel(network_file,
sheet_name=sheet, skiprows=1)
            sub_df = sub_df.dropna(subset=["Site Name", "Site
Code"]).copy()
            sub_df["Cleaned Site"] = sub_df["Site
Name"].apply(clean_name)
            substations.append(sub_df[["Site Name", "Site
Code", "Cleaned Site"]])
        sub_df = pd.concat(substations, ignore_index=True)
        sub_df = sub_df.drop_duplicates(subset=["Cleaned
Site"])
        return sub_df

    # Load TEC Register
    gen_df = pd.read_excel(gen_file, sheet_name=gen_sheet,
skiprows=1)
    gen_df = gen_df.dropna(subset=["Project Status", "HOST
TO"])
    gen_df["Project Status"] = gen_df["Project
Status"].str.strip().str.lower()
    gen_df["HOST TO"] = gen_df["HOST
TO"].str.strip().str.upper()

    # Filter only built NGET generators
    gen_df = gen_df[
        (gen_df["Project Status"] == "built") &
        (gen_df["HOST TO"] == "NGET")
    ].copy()
```

```
    # Clean names and map categories
    gen_df["Cleaned Site"] = gen_df["Connection
Site"].apply(clean_name)
    gen_df["LCOE Category"] = gen_df["Plant
Type"].apply(map_to_lcoe_category)

    # Load substations and match
    sub_df = load_substations()
    site_names_clean = sub_df["Cleaned Site"].tolist()
    gen_df["Matched Site"], gen_df["Match Score"] =
zip(*gen_df["Cleaned Site"].map(
        lambda x: get_best_match(x, site_names_clean)
    ))

    # Merge generators with substations
    matched_df = pd.merge(
        gen_df,
        sub_df,
        left_on="Matched Site",
        right_on="Cleaned Site",
        how="left",
        suffixes=("", "_sub")
    )

    # Keep only needed columns — each row is ONE generator
    final_matched = matched_df[[
        "Connection Site", "Site Code", "Site Name",
        "MW Connected", "Match Score", "LCOE Category"
    ]].copy()

    # Fixed costs €/MWh
    fixed_costs = {
        "Wind": 10,
        "Solar PV": 15,
```

```
        "Nuclear": 20,
        "Hydro": 40,
        "Pumped Storage": 65,
        "CCGT": 50,
        "OCGT": 60,
        "CHP": 90,
        "Oil": 55,
        "Biomass": 35,
        "Other": 70,
        "Coal": 80,
        "Battery Storage": 65
    }

    final_matched["Fixed Cost"] = final_matched["LCOE
Category"].map(fixed_costs).fillna(1000)

    # Clear old poly_cost
    if not net.poly_cost.empty:
        net.poly_cost.drop(net.poly_cost.index, inplace=True)

    # Loop through each generator row
    for _, row in final_matched.iterrows():
        site_code = row["Site Code"]
        total_mw = row["MW Connected"]
        fixed_cost = row["Fixed Cost"]
        gen_type = row["LCOE Category"]

        if site_code not in substation_group:
            continue

        buses = substation_group[site_code]
        if not buses:
            continue

        mw_per_bus = total_mw / len(buses)
```

```
        for bus_name in buses:
            if bus_name in NGET_bus_lookup:
                bus_idx = NGET_bus_lookup[bus_name]

                # Create generator
                pp.create_gen(
                    net, bus=bus_idx, p_mw=0.0, min_p_mw=0.0,
                    max_p_mw=mw_per_bus, name=gen_type,
controllable=True
                )

                # Add cost
                gen_idx = net.gen.index[-1]
                pp.create_poly_cost(
                    net, element=gen_idx, et='gen',
                    cp0_eur=0.0,
                    cp1_eur_per_mw=fixed_cost,
                    cp2_eur_per_mw2=0.0
                )

    print("Generator creation complete.")
```

C.8 run_dc.py

```
import pandapower.topology as ppt
import pandapower as pp
import warnings

warnings.filterwarnings('ignore')

def run_dcopf(net):
    # Reduce the generation of Nuclear generators by half
    nuclear_gens =
net.gen[net.gen['name'].str.contains('Nuclear',
case=False)].index
    net.gen.loc[nuclear_gens, 'max_p_mw'] *= 0.5



    # Create the networkx graph
    graph = ppt.create_nxgraph(net)

    # Find all islands (connected components)
    islands = list(ppt.connected_components(graph))

    if not islands:
        raise ValueError("No islands found in the network.")

    # Find the largest island by bus count
    largest_island = max(islands, key=len)

    # Buses to drop = all except the largest island
    buses_to_drop = list(set(net.bus.index) -
set(largest_island))

    if buses_to_drop:
        pp.drop_buses(net, buses_to_drop)

    # Ensure we still have a slack or generators
```

```
    if net.ext_grid.empty and net.gen.empty:
        raise ValueError("No slack/ext_grid or generator in
the largest island; cannot run power flow.")

    # --- Run DC Power Flow ---
    pp.rundcpp(net)
    print('DCPF converged:', net.converged)

    # --- Run DC Optimal Power Flow ---
    try:
        pp.rundcopp(net)
        print('DCOPF converged:', net.OPF_converged)
    except Exception as e:
        print('DCOPF failed to converge')
        print('Error:', e)

    return net
```

C.9 plotting.py

```
import pandapower.topology as top
import networkx as nx
import matplotlib.pyplot as plt
import os




def plot_network(net):
    # Network topology
    # Create the graph from the pandapower network
    G = top.create_nxgraph(net, respect_switches=True)

    # Use spring layout with tuned spacing (k increases
spacing between nodes)
    pos = nx.spring_layout(G, k=4.0, iterations=300, seed=42)

    # Color nodes by type: green = generator, orange = load,
lightblue = regular bus
    node_colors = []
    for node in G.nodes:
        if node in net.gen.bus.values:
            node_colors.append("navy")        # generators →
dark navy blue
        elif node in net.load.bus.values:
            node_colors.append("darkred")   # loads → dark red
        elif node in net.ext_grid.bus.values:
            node_colors.append("black")       # external grid
→ black
        else:
            node_colors.append("lightblue")

    # Optional: Label a few key buses (e.g., ext_grid or
largest gen)
    labels = {}
```

```
    for node in G.nodes:
        if node in net.ext_grid.bus.values or node in
net.gen.bus.value_counts().head(5).index:
            labels[node] = f"Bus {node}"

    # Plotting
    plt.figure(figsize=(24, 20))  # Large canvas
    nx.draw_networkx_nodes(G, pos,
                          node_color=node_colors,
                          node_size=80,
                          alpha=0.9)
    nx.draw_networkx_edges(G, pos,
                          edge_color="gray",
                          alpha=0.5,
                          width=1)
    nx.draw_networkx_labels(G, pos, labels,
                              font_size=8,
                              font_color="black")

    plt.title("PandaPower Network Topology (Well-Spaced, High
Detail)", fontsize=18)
    plt.axis("off")
    plt.tight_layout()

    # Get path to results folder (one level above "utility")
    results_dir = os.path.join(os.path.dirname(__file__),
"..", "results")
    os.makedirs(results_dir, exist_ok=True)

    # Save high-resolution image inside results
    plt.savefig(os.path.join(results_dir,
"grid_topology.png"), dpi=400)
    plt.show()
```

C.10 generation_summary.py

```
# gen_summary.py

def generation_summary(net):
    gen_types = [
    "Wind",
    "Solar PV",
    "Nuclear",
    "Hydro",
    "Pumped Storage",
    "CCGT",
    "OCGT",
    "CHP",
    "Biomass",
    "Other",
    "Coal",
    "Battery Storage",
    "BritNed",
    "East-West",
    "Nemo Link",
    "IFA-2",
    "IFA",
    "North Sea Link",
    "ElecLink",
    "Viking Link",
    "Greenlink"
    ]

    external_grid_mw = net.res_ext_grid["p_mw"].sum()
    total_generation_mw = net.res_gen['p_mw'].sum() +
external_grid_mw
    total_load_mw = net.res_load["p_mw"].sum()
    generation_by_type = {gen_type: 0.0 for gen_type in
gen_types}
    generation_by_type['Unknown'] = 0.0
```

```
    for idx, gen in net.gen.iterrows():
        gen_name = gen['name']
        gen_output = net.res_gen.at[idx, 'p_mw']

        matched_type = None
        for gen_type in gen_types:
            if gen_type.lower() in gen_name.lower():
                matched_type = gen_type
                break

        if matched_type:
            generation_by_type[matched_type] += gen_output
        else:
            generation_by_type['Unknown'] += gen_output

    print(f"Total load in network: {total_load_mw:.2f} MW")
    print(f"Total generation in network:
{total_generation_mw:.2f} MW")
    print("Generation by type:")
    print(f"  EXT_GRID: {external_grid_mw:.2f} MW" )
    for gen_type, total in generation_by_type.items():
        print(f"  {gen_type}: {total:.2f} MW")
```

C.11 user_input.py

```
# --- User Inputs ---

network_file = "ETYS_documents/ETYS_B.xlsx"
demand_file = "ETYS_documents/ETYS_G.xlsx"
generator_file = "ETYS_documents/ETYS_F.xlsx"

bus_sheet = ["B-2-1c", "B-3-1c"]
substation_sheet = ["B-1-1a", "B-1-1b", "B-1-1c", "B-1-1d"]
transformer_sheet = "B-3-1c"
line_sheet = "B-2-1c"
load_sheet = "demand data 2023"
gen_sheet = "TEC Register"
```

C.12 spt.py

```
import pandapower as pp

def create_spt_assets(net, SPT_BUS, total_SHE_load,
total_SPT_load):

    SPT_LOAD = total_SHE_load + total_SPT_load

    OFTO_GEN = 4388.4

    SPT_GEN = {
    "CCGT": 1200,
    "CHP": 120,
    "Battery Storage": 97.95,
    "Hydro": 951.4,
    "Nuclear": 1270,
    "Pumped Storage": 740,
    "Wind": 8154.6
    }

    fixed_costs = {
        "Wind": 10,
        "Solar PV": 15,
        "Nuclear": 20,
        "Hydro": 40,
        "Pumped Storage": 65,
        "CCGT": 50,
        "OCGT": 60,
        "CHP": 90,
        "Oil": 55,
        "Biomass": 35,
        "Other": 70,
        "Coal": 80,
        "Battery Storage": 65
    }
```

```
    # Slack Bus
    EXT_GRID = pp.create_ext_grid(net, SPT_BUS, vm_pu=1,
name="SPT BUS")
    net.ext_grid.at[EXT_GRID, "max_p_mw"] = 0  # max import
    net.ext_grid.at[EXT_GRID, "min_p_mw"] = 0


    # SPT load creation
    pp.create_load(net, SPT_BUS, p_mw=SPT_LOAD,
controllable=False)

    # offshore wind generation
    OFTO = pp.create_gen(net, SPT_BUS, p_mw=OFTO_GEN,
min_p_mw=0.0, max_p_mw=OFTO_GEN, name="Wind",
controllable=False)
    pp.create_poly_cost(net, element=OFTO, et="gen",
cp1_eur_per_mw=10)

    for gen_type, capacity_mw in SPT_GEN.items():
        pp.create_gen(
            net,
            SPT_BUS,
            p_mw=0,
            min_p_mw=0,
            max_p_mw=capacity_mw,
            vm_pu=1.0,
            name=gen_type,
            controllable=True
        )

        # Add cost to OPF
        pp.create_poly_cost(
            net,
```

```
        element=len(net.gen) - 1,
        et="gen",
        cp1_eur_per_mw=fixed_costs[gen_type]
    )

print(SPT_LOAD, "SPT LOAD")
```

C.13 interconnectors.py

```
import pandapower as pp
import math


def convert_mva_to_ka(mva_rating):
    return ((mva_rating * 1_000_000) / ( math.sqrt(3) *
400_000)) / 1000

# add scaling factor for the function.

def create_interconnectors(net, NGET_bus_lookup,
mode="import"):
    # Define buses
    france_1 = pp.create_bus(net, vn_kv=400, name="France")
    france_2 = pp.create_bus(net, vn_kv=400, name="France")
    france_3 = pp.create_bus(net, vn_kv=400, name="France")
    netherlands = pp.create_bus(net, vn_kv=400,
name="Netherlands")
    belgium = pp.create_bus(net, vn_kv=400, name="Belgium")
    norway = pp.create_bus(net, vn_kv=400, name="Norway")
    denmark = pp.create_bus(net, vn_kv=400, name="Denmark")
    ireland_1 = pp.create_bus(net, vn_kv=400, name="Ireland")
    ireland_2 = pp.create_bus(net, vn_kv=400, name="Ireland")

    interconnectors = {
    "IFA": {
        "capacity_mw": 2000,
        "from_bus": "SELL",
        "to_bus": france_1,
        "length_km": 73,
        "mode": "import",
        "mw_flow": 1504
    },
    "BritNed": {
        "capacity_mw": 1000,
```

```
        "from_bus": "GRAI",
        "to_bus": netherlands,
        "length_km": 260,
        "mode": "import",
        "mw_flow": 783
    },
    "East-West": {
        "capacity_mw": 500,
        "from_bus": "FLIB",
        "to_bus": ireland_1,
        "length_km": 261,
        "mode": "export",
        "mw_flow": 500
    },
    "Nemo Link": {
        "capacity_mw": 1000,
        "from_bus": "RICH",
        "to_bus": belgium,
        "length_km": 140,
        "mode": "import",
        "mw_flow": 23
    },
    "IFA-2": {
        "capacity_mw": 1000,
        "from_bus": "CHIL",
        "to_bus": france_2,
        "length_km": 204,
        "mode": "import",
        "mw_flow": 992
    },
    "North Sea Link": {
        "capacity_mw": 1400,
        "from_bus": "BLYT",
        "to_bus": norway,
        "length_km": 720,
```

```
        "mode": "import",
        "mw_flow": 1399
    },
    "ElecLink": {
        "capacity_mw": 1000,
        "from_bus": "SELL",
        "to_bus": france_3,
        "length_km": 51,
        "mode": "import",
        "mw_flow": 996
    },
    "Viking Link": {
        "capacity_mw": 1400,
        "from_bus": "BICF",
        "to_bus": denmark,
        "length_km": 765,
        "mode": "import",
        "mw_flow": 368
    },
    "Greenlink": {
        "capacity_mw": 500,
        "from_bus": "PEMB",
        "to_bus": ireland_2,
        "length_km": 190,
        "mode": "export",
        "mw_flow": 452
    },
    }

    for name, data in interconnectors.items():
        bus_name = data["to_bus"]
        max_p_mw = data["capacity_mw"]

        if mode == "import":
            # Interconnector acts as a generator importing
```

```
power to your network
            pp.create_gen(net, bus=bus_name, p_mw=0.0,
min_p_mw=0.0,
                    max_p_mw=max_p_mw, vm_pu=1, name=name,
controllable=True)
            pp.create_poly_cost(
                    net, element=bus_name, et='gen',
                    cp0_eur=0.0,
                    cp1_eur_per_mw=30,
                    cp2_eur_per_mw2=0.0
            )
        elif mode == "export":
            # Interconnector acts as a load exporting power
from your network
            pp.create_load(net, bus=bus_name, p_mw=max_p_mw,
name=name)

        elif mode == "manual":
            ic_mode = data["mode"]
            p_mw = data["mw_flow"]

            if ic_mode == "import":
                # Interconnector acts as a generator importing
power to your network
                pp.create_gen(net, bus=bus_name, p_mw=p_mw,
min_p_mw=0.0,
                        max_p_mw=max_p_mw, vm_pu=1, name=name,
controllable=False)
                pp.create_poly_cost(
                        net, element=bus_name, et='gen',
                        cp0_eur=0.0,
                        cp1_eur_per_mw=30,
                        cp2_eur_per_mw2=0.0
                )
            elif ic_mode == "export":
```

```
                # Interconnector acts as a load exporting
power from your network
                pp.create_load(net, bus=bus_name,
p_mw=max_p_mw, name=name)

        else:
            raise ValueError("Invalid mode. Use 'import' or
'export'.")


    for name, data in interconnectors.items():
        sub_prefix = data["from_bus"].lower()
        matching_bus = None

        for key, value in NGET_bus_lookup.items():
            if key.lower().startswith(sub_prefix):
                matching_bus = value
                break

        if matching_bus is None:
            raise KeyError(f"No matching bus found for prefix
{sub_prefix}")

        from_bus = matching_bus
        to_bus = data["to_bus"]
        length_km = data["length_km"]
        capacity_mw = data["capacity_mw"]
        ka_rating = convert_mva_to_ka(capacity_mw)


        pp.create_std_type(
            net,
            {
                "c_nf_per_km": 0,
                "r_ohm_per_km": 0,
```

```
            "x_ohm_per_km": 0.05,
            "max_i_ka": ka_rating,
            "g_us_per_km": 0,
            "type": "cs"
        },
        name=f"std_{name}",
        element="line"
    )

    pp.create_line(
        net,
        from_bus=from_bus,
        to_bus=to_bus,
        length_km=length_km,
        std_type=f"std_{name}",  # Custom standard type
        name="Interconnector",
        max_loading_percent=100
    )

print("Interconnector creation complete.")
```

C.14 security.py

```
# security.py
import pandas as pd
import pandapower as pp


def n1_security(net, step=0.01, max_scale=2.0,
include_impedances=True, loading_limit=100.0):
    """
    Incrementally increase load and run N-1 security screening
using DC OPF (pp.rundcopp).
    Stops at the first step where the system becomes insecure.

    Parameters
    ----------
    net : pandapowerNet
        Your pandapower network.
    step : float
        Load increment per step (e.g., 0.01 = +1%).
    max_scale : float
        Max multiplier to try (safety cap).
    include_impedances : bool
        If True, also test outages of impedance elements
(e.g., trafos modelled as impedance).
    loading_limit : float
        Loading percent threshold treated as a violation
(default 100%).

    Returns
    -------
    secure_scale : float
        Last multiplier at which all N-1 contingencies
remained secure.
    violations_df : pandas.DataFrame
        Violations at the first insecure step. Columns:
element_type, element_index, max_loading_percent.
```

```
        Empty DataFrame if no violations up to max_scale.
    """
    # Cache base load so scaling is absolute each step (not
compounding)
    base_load = net.load["p_mw"].copy()

    # Build contingency list: lines (and optionally
impedances)
    contingencies = [("line", i) for i in net.line.index]
    if include_impedances and hasattr(net, "impedance") and
len(net.impedance):
        contingencies += [("impedance", i) for i in
net.impedance.index]

    scale = 1.0
    secure_scale = 1.0

    while scale + step <= max_scale + 1e-12:
        scale = round(scale + step, 8)  # avoid float creep
        # Absolute scaling from base profile
        net.load["p_mw"] = base_load * scale

        # Track violations at this scale across all
contingencies
        violations = []

        for etype, idx in contingencies:
            # Toggle the element out
            if etype == "line":
                old_state = net.line.at[idx, "in_service"]
                net.line.at[idx, "in_service"] = False
            elif etype == "impedance":
                old_state = net.impedance.at[idx,
"in_service"]
                net.impedance.at[idx, "in_service"] = False
```

```
            else:
                continue  # unknown type (shouldn't happen)

            try:
                # Run DC OPF
                pp.rundcopp(net)

                # Evaluate line loadings (post-contingency)
                if hasattr(net, "res_line") and
len(net.res_line):
                    max_loading =
float(net.res_line["loading_percent"].max())
                else:
                    # If results missing, consider it a
failure
                    max_loading = float("inf")

                if max_loading >= loading_limit:
                    violations.append({
                        "element_type": etype,
                        "element_index": idx,
                        "max_loading_percent": max_loading
                    })

            except Exception as e:
                # Any solver failure is treated as insecurity
                violations.append({
                    "element_type": etype,
                    "element_index": idx,
                    "max_loading_percent": float("inf"),
                    "error": str(e)
                })

            finally:
                # Restore the element
```

```
                if etype == "line":
                    net.line.at[idx, "in_service"] = old_state
                elif etype == "impedance":
                    net.impedance.at[idx, "in_service"] =
old_state

        # If any contingency violates, we’re at the first
insecure step
        if violations:
            return secure_scale, pd.DataFrame(violations)

        # Otherwise, this step is secure
        secure_scale = scale

    # No violation up to max_scale
    return secure_scale, pd.DataFrame()
```

C.15 get_results.py

```
import pandas as pd
import pandapower as pp
import os


def get_results(net):
    # Ensure the results folder exists
    os.makedirs("results", exist_ok=True)

    # Save Network for reference
    pp.to_excel(net, os.path.join("results", "network.xlsx"))


    # --- Save Results ---
    with pd.ExcelWriter(os.path.join("results",
"dc_opf_results_winter_scenario.xlsx")) as writer:
        net.res_bus.to_excel(writer, sheet_name="Bus Results")
        net.res_line.to_excel(writer, sheet_name="Line
Results")
        net.res_gen.to_excel(writer, sheet_name="Generator
Results")
        net.res_load.to_excel(writer, sheet_name="Load
Results")
        net.res_ext_grid.to_excel(writer, sheet_name="External
Grid Results")
```

C.16 Folder Structure

```
project_root/
│
├── ETYS_documents/           # Input datasets from NESO ETYS appendix
│   ├── ETYS_B.xlsx           # Appendix B – Network data
│   ├── ETYS_F.xlsx           # Appendix F – Generation data
│   └── ETYS_G.xlsx           # Appendix G – Demand data
│
├── network.py             # Main script that builds and runs the transmission model
├── user_input.py          # Defines user inputs (file paths, sheet names, etc.)
│
├── elements/              # Network element creation modules
│   ├── buses.py           # Functions to create buses from substation data
│   ├── substations.py     # Substation definitions and lookup
│   ├── transformers.py    # Transformer creation and impedance modelling
│   ├── loads.py           # Load allocation and bus grouping
│   ├── lines.py           # Transmission line creation and parameter conversion
│   └── generators.py      # Generator allocation and cost assignment
│
├── validation_elements/   # Components for model validation
│   ├── interconnectors.py # Creates interconnectors (IFA, BritNed, NSL, etc.)
│   └── spt.py             # Reduced Scottish Power Transmission representation
│
├── utilities/             # Helper modules for running and analysing studies
│   ├── run_dc.py          # DC Optimal Power Flow execution
│   ├── plotting.py        # Functions for network plotting/visualisation
│   ├── generation_summary.py # Summarises generation dispatch results
│   ├── security.py        # N-1 security assessment functions
│   └── get_results.py     # Exports and processes OPF/network results
```