SET3065 Intelligent Electrical Power Grids (Theory)

Responsible professors:

Peter Palensky

Jose Rueda

Pedro Vergara Barrios

Jochen Cremer

Contents

1. Digital energy

This chapter concerns with the evolution of power systems and the development of smart grid technologies, including SCADA systems, digital substations, and synchronized measurement tools like PMUs and PDCs.

The learning objectives are:

(i) Describe the evolution and challenges in modern power systems.

(ii) Identify key components and technologies of smart grid infrastructure.

(iii) Evaluate the roles of SCADA systems and digital substations.

(iv) Explain the function and importance of synchronized measurement tools such as PMUs and PDCs.

1.1 Power Systems

A power system is a network of electrical components used to generate, transmit, distribute, and consume electric power. It includes power generation units, transmission lines, substations, distribution networks, and the end-users or loads. The main goal of a power system is to deliver electricity reliably, efficiently, and safely from producers to consumers, while maintaining stability and quality of supply under varying demand and operating conditions.

1.1.1 Challenges and Roadblocks

Traditional power grids were designed with centralized generation in mind, using large-scale fossil fuel or nuclear plants to deliver power over long distances. These systems operate with a relatively predictable and unidirectional flow of energy from generation to consumer. However, the rise of renewable energy sources such as solar and wind presents a new set of challenges. While renewable energy offers environmental and economic benefits, it also introduces issues like intermittency, geographic dependency, and difficulties integrating with existing grid infrastructure. Achieving a 100% renewable grid faces several key roadblocks. Existing infrastructure often lacks the technology to handle the variability and decentralization of renewables. Without adequate storage and flexibility, power systems struggle to balance supply and demand, especially during peak or low-generation periods. Transmission networks may become overloaded, and maintaining voltage and frequency stability is harder due to the absence of inertia from traditional generators. Additionally, resilience to blackouts and cyber threats, alongside the high cost of infrastructure upgrades, presents major technical and economic barriers.

1.1.2 Future Infrastructure

The future of power systems lies in a hybrid infrastructure that balances centralized, decentralized, and distributed models. Centralized generation will still play a role, especially for large-scale, consistent energy production. However, the growth of decentralized and distributed energy resources (DERs), such as rooftop solar panels, wind turbines, and battery storage, is reshaping the grid into a more flexible and resilient network. Decentralized systems like microgrids enhance local energy independence and reduce reliance on distant generation and transmission. Distributed energy allows consumers to become prosumers, both producing and consuming electricity. To support this evolution, investments in storage technologies, flexible demand management, and advanced grid control systems are essential. Regulatory support and innovative market structures will also be crucial in enabling efficient integration and coordination across all levels of the grid.

1.1.3 Digital Transformation of Smart Grids

Digital transformation is the backbone of this future-ready grid. By incorporating technologies such as artificial intelligence, and real-time data analytics, smart grids allow for enhanced visibility, control, and automation. Two-way communication enables utilities to interact directly with consumer devices, manage demand, and optimize power flows dynamically. Smart sensors, advanced metering infrastructure, and automated control systems help detect faults, respond to fluctuations, and reduce blackouts. These technologies also facilitate the integration of renewable energy and storage by forecasting generation and consumption patterns. As more devices connect to the grid, robust cybersecurity becomes essential to protect system integrity. Overall, digital transformation enables a smarter, cleaner, and more responsive power system capable of adapting to the evolving energy landscape.

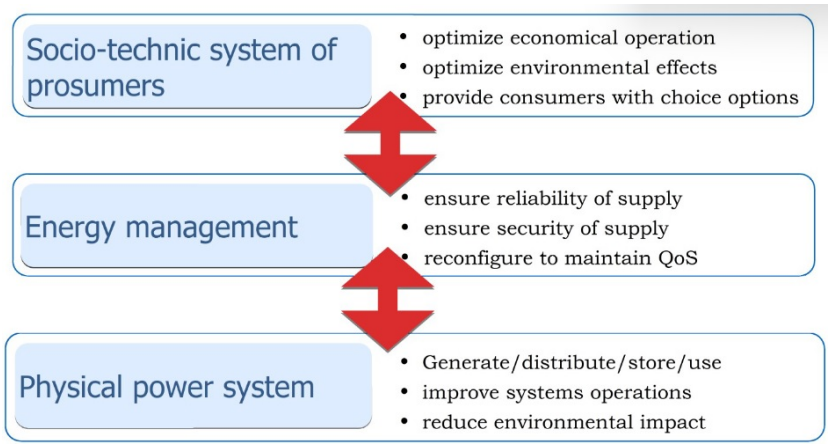


Figure 1.1: Energy System innovation is driven by the interests of the different parties that partake in the operation of the grid

1.2 Modeling and Simulation of Smart Grids

1.2.1 The Grid

The electrical grid is operated by several parties that work in a coordinated manner, each taking care of specific tasks, as shown in Figure 1.1.

Monitoring solutions are needed in order to know the state of the grid at any given moment. Grid monitoring solutions allow grid operators to monitor grid loading and operating frequency, and are able to discern data on load behavior and demand response as well as to aggregate virtual storages. They are therefore fundamental tools in a grid operator's ability to coordinate with energy suppliers and to shed or shift load as necessary.

Figure 1.2: IRON Box

The IRON Box, depicted in Figure 1.2 and designed by Prof. Peter Palensky, was used for demand management in cooling houses for vegetables. The thermal inertia of a cooling house is very large, allowing for smart management of the energy demand (i.e. cooling to below target temperature when the market price of electricity is low and letting the temperature slowly rise when it is high).

The system behavior of a grid can be analyzed in several ways. One such way is to plot a load vs. frequency graph. A load vs frequency curve with a **positive** gradient, as seen in Figure 1.3, is an indicator of power-grid friendly behavior, as a positive frequency deviation is correlated with more generation than demand, and a negative deviation correlates with a generation deficit.

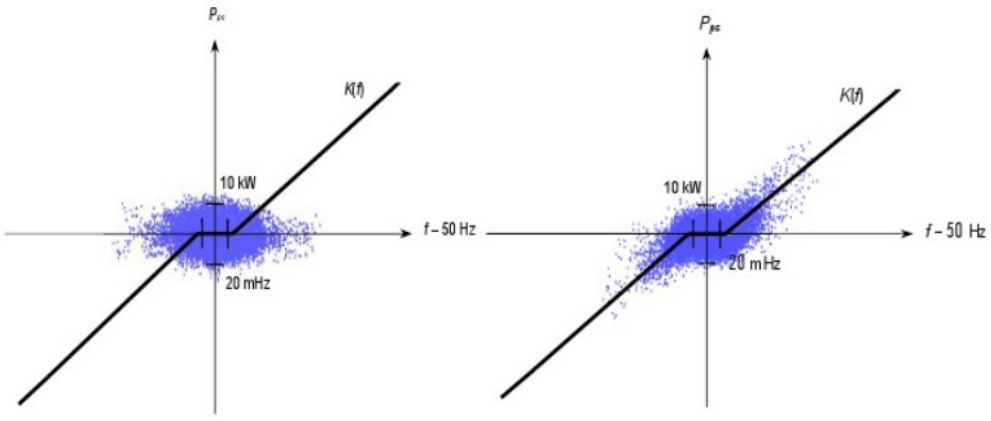


Figure 1.3: Example of a power-grid friendly system behavior (on the right graph)

As the amount of elements with time-dependent behavior increases, the complexity of a system too increases rapidly. Such systems is not trivial to calculate or even simulate in a lab.

1.2.2 Physical models

Physical models can be represented in several manners:

- **Analytic:** No solver, no interactions.
- **Causal:** Unidirectional flow of information, uses transfer functions, good for control systems. Matlab and Ptolemy fall in this category.
- **Acausal:** Bidirectional flow of information, equation-based, good for physical modelling. Modellica and Symscape fall in this category.

In order to create a physical model, an engineer must translate reality into a system that can be represented in the model's available elements.

Physical models are heterogeneous even within the time domain, since the behaviour of a system usually depends on the timescale.

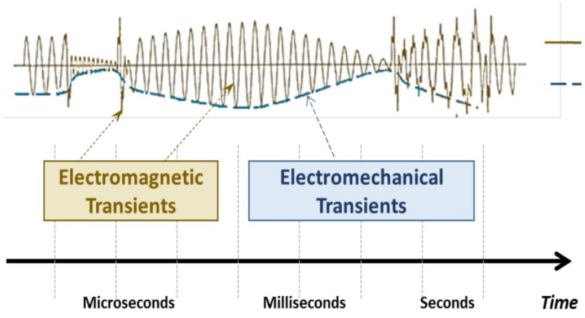


Figure 1.4: Example of electromagnetic and electromechanical transients in time domain

In the example shown in Figure 1.4, an EMT model is the closest to representing reality, but is quite complex and computationally expensive to calculate. Using a TS model in the phasor domain, is simpler and easier to calculate, although it cannot represent switching

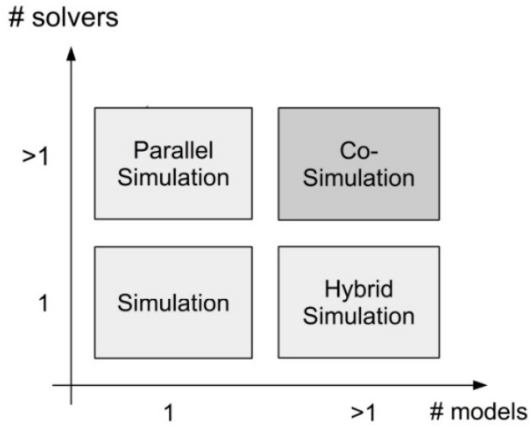


Figure 1.5: Number of solvers vs. number of models, showing the highly location of co-simulation

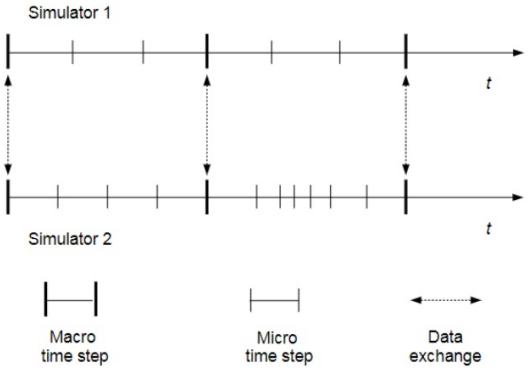


Figure 1.6: Graphical description of checkpoint synchronization for co-simulation

accurately. Steady state models are the simplest, but as the name suggests, they cannot represent dynamic systems but only a single snapshot.

1.2.3 Co-simulation

In co-simulation, dedicated models are used for each domain, each with its own model and solver, as seen in Figure 1.5. Coupling them is then done through a ”simulation master” algorithm which initializes the individual model simulators, exchanges the variables, and keeps the system synchronous.

Synchronisation is attained by using checkpoints, as shown in Figure 1.6, at which each solver will stop and report to the master algorithm, which then will wait until all the solver have caught up to that point and then communicate them with each other and restart them as necessary. This method is lossy, since communication is not continuous.

There are two coupling methods which can be used, illustrated in Figure 1.7. Explicit coupling exchanges solver data at every checkpoint synchronously, while implicit coupling iterates on each checkpoint step until all parameters in the system converge within a given tolerance. In both cases, the error is related to the amount of steps used or the distance between each step. This is illustrated in Figure 1.7.

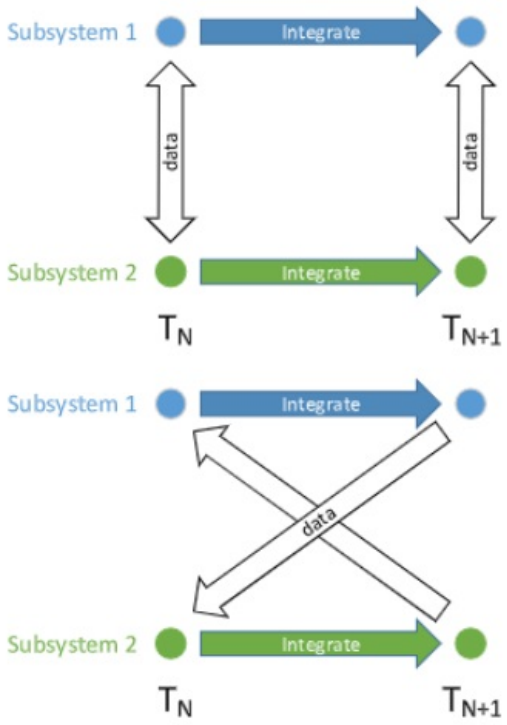


Figure 1.7: Coupling principles for co-simulation

1.2.4 Co-simulation software

Co-simulation has multiple focuses, it allows each model to be built by a specialist in an environment that is tailored specifically for it, therefore allowing for maximum accuracy. On the downside, synchronisation between the different models implies a large computational load that makes co-simulation relatively slow. Below are brief descriptions of the most common co-simulation software.

DIgSILENT PowerFactory

PowerFactory, shown through screenshots in Figure 1.8, is an extremely powerful tool for the modelling of electrical systems that is capable of simulating different models. It has a large database of components that can be quickly imported into your models. It can use static simulations, quasi-static simulations which represent the system statically every time-step, or dynamic simulations usually in the phasor domain (Transient Stability, Root Mean Square) or in Electromagnetic Transients.

Complicated models can be co-simulated inside PowerFactory by for example simulating fast-reacting components in Electromagnetic Transient mode and the rest of the system in phasor mode, and then coupling them together via FMI's for example.

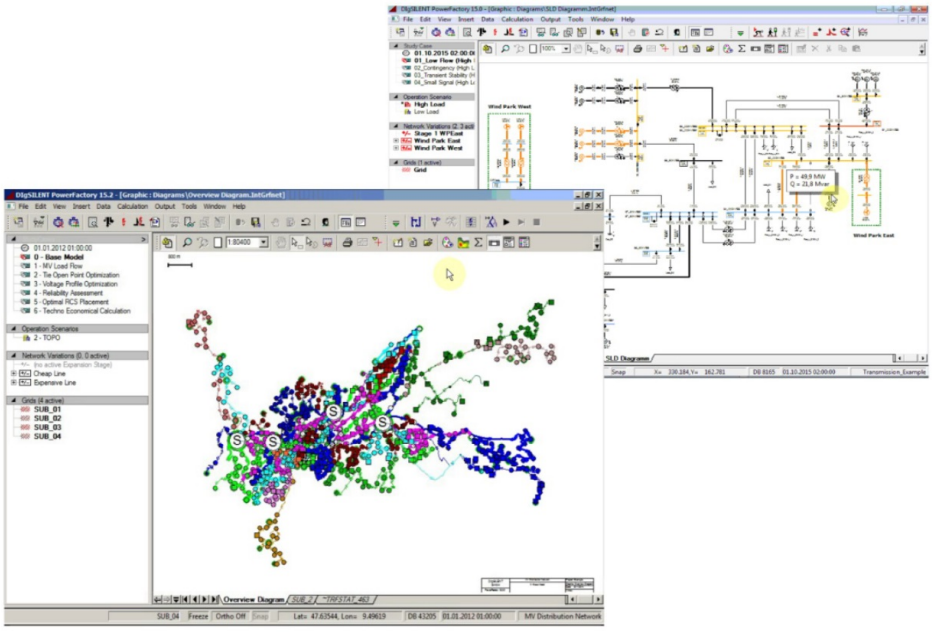


Figure 1.8: Screenshots from PowerFactory

Matpower

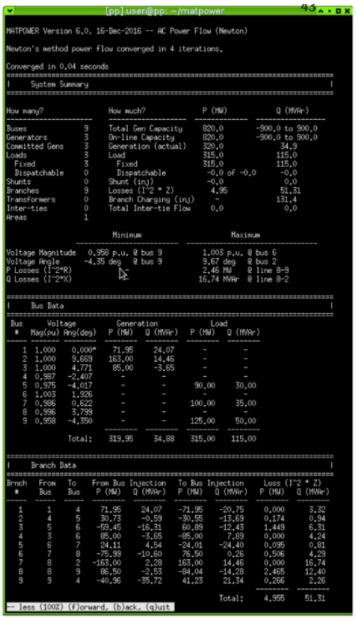


Figure 1.9: Example of Matpower Command Prompt Interface

Matpower, shown in Figure 1.9 is a free tool that can only perform power flow simulation. You describe the grid in a static manner and it performs the calculations.

Pandapower

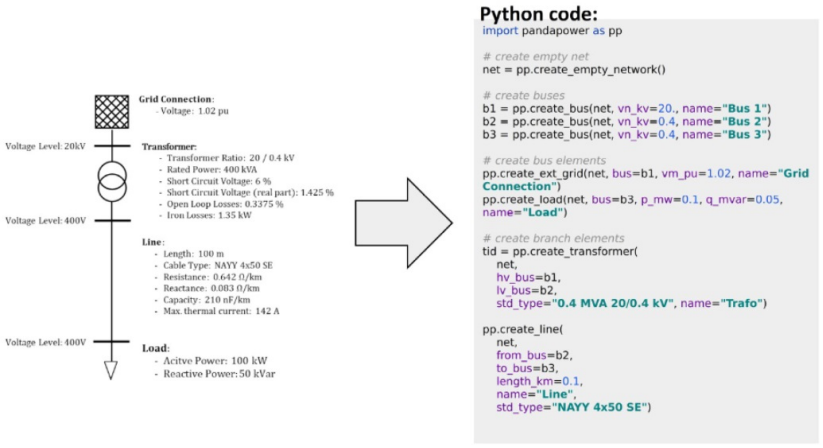


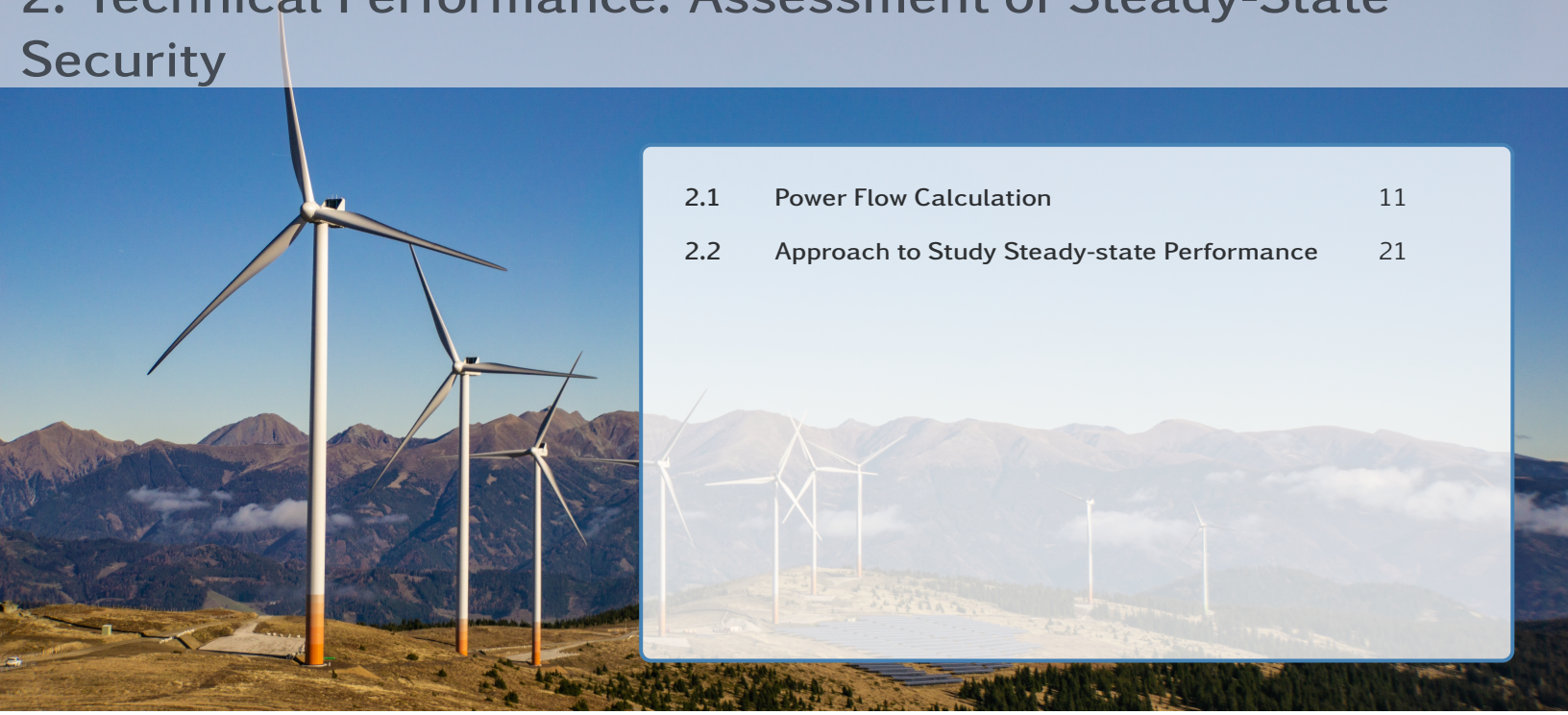
Figure 1.10: Example of system flowchart and equivalent Python code for Pandapower

Pandapower, as seen on Figure 1.10, is based on Python, and in this course is mainly used for powerflow calculations. You set up by describing the grid topology and parameters and then it can perform power flow calculations and power flow optimization.

Bottom line

Heterogeneous systems are not good at multi-disciplinary models, therefore co-simulation is the way to go. It is still best to use as few models as possible to prevent excessive problems with versioning and future work.

2. Technical Performance: Assessment of Steady-State Security



This chapter concerns with the steady-state analysis of power systems using power flow equations derived from the network admittance matrix Y_{bus} and solved through iterative techniques such as the Newton-Raphson method.

The learning objectives are:

(i) Apply power flow calculation for steady-state simulation.

(ii) Implement steady-state performance metrics.

(iii) Evaluate power system steady-state performance.

2.1 Power Flow Calculation

In the context of power systems, steady-state security refers to the ability of the system to operate within acceptable limits following normal operating conditions or minor disturbances, such as small changes in load or generation. Ensuring steady-state security is critical for maintaining reliable service and avoiding equipment overloading or voltage instability. One of the foundational tools for assessing steady-state security is power flow analysis, which enables the examination of voltage profiles, power transfers, and system losses under given operating scenarios. The following section outlines the basic inputs and objectives of power

flow studies, setting the stage for further analysis of system behavior and performance under various operational conditions.

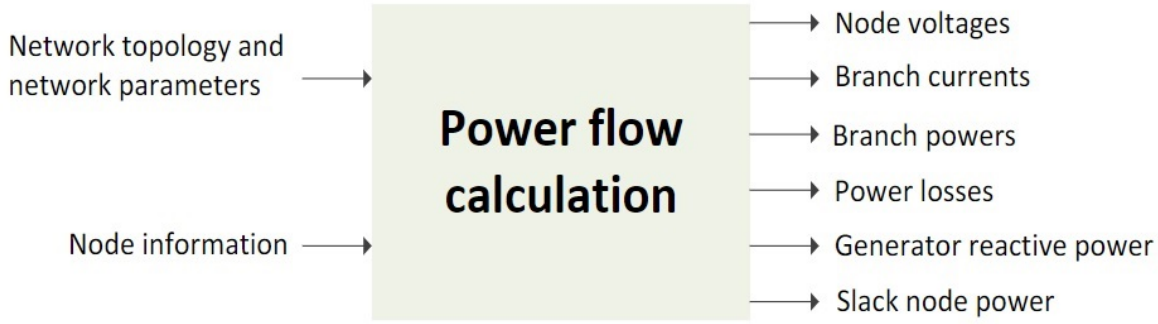


Figure 2.1: Schematic of steady-state power flow calculation

Generally speaking, power flow (or load flow) analysis, as illustrated in Figure 2.1, involves solving for unknown variables in a power system based on a defined set of input parameters, including:

- The active power generation of all generators, excluding the slack generator, as well as the system load demand.
- The reactive power consumption of loads.
- Specified voltage magnitudes at generator buses with known active power output.
- The voltage magnitude and phase angle at the slack (or reference) bus.

Power flow calculations are essential for the reliable and efficient operation of electrical power systems. They provide insight into the steady-state behavior of the network, enabling operators and engineers to make informed decisions. Key applications include state estimation for real-time monitoring and control, assessing the feasibility and efficiency of power dispatch strategies, and evaluating system response under various contingencies (such as line or generator outages). Accurate power flow analysis supports operational planning, stability assessments, and the integration of renewable energy sources into the grid.

2.1.1 YBUS: Network Admittance Matrix

Generalizing for any power system, the network admittance matrix, denoted as Y_{bus}, defines the relationship between the nodal voltages vector, U, and the nodal currents vector, I, using the equation:

$$\mathbf{I}_{[N\times 1]} = \mathbf{Y}_{\text{bus}\,[N\times N]} \cdot \mathbf{U}_{[N\times 1]}, \tag{2.1}$$

where N is the total number of network buses (nodes), I is the nodal currents vector representing the current injected into each bus, U is the nodal voltages vector representing the voltage at each bus and Y_{bus} is the network admittance matrix describing the electrical connectivity and admittances between nodes. The Y_{bus} matrix is square with dimensions N × N, typically symmetric for passive systems. The diagonal elements correspond to the total

self-admittance at each node, while the off-diagonal elements represent the negative mutual admittance between connected nodes.

There are two common approaches for constructing the Y_{bus} matrix:

- **Manual Inspection**
  It is a straightforward method for constructing the Y_{bus} matrix, particularly effective in small power systems where the network is simple and easily understood. The process involves two basic rules for determining the elements of the matrix.

  – For the diagonal elements, Y_{ii}, the value is the sum of all the admittances connected to node i. In other words, the diagonal element represents the total admittance at node i, which is the sum of the admittances of all branches connected to that node.

  $$\bar{Y}_{ii} = \sum \bar{Y}_{ik}$$

  – For the off-diagonal elements, Y_{ij} (for $i \neq j$), the value is the negative of the admittance between node i and node j.

  $$\bar{Y}_{ij} = -\bar{Y}_{ij}$$

  This reflects the fact that the admittance between two nodes is shared, and each off-diagonal element represents the mutual admittance between those nodes. This method is intuitive and easy to apply in small networks, however, it can become increasingly time-consuming for larger systems, where more efficient computational techniques are preferred. Using this rule, the Y_{bus} matrix is assembled row-by-row, applying Kirchhoff's Current Law (KCL) at each node by summing incoming and outgoing admittances.

  To understand how the Y_{bus} matrix governs the steady-state behavior of power systems, we start by writing the individual nodal equations. Each equation expresses the current injected into a bus as a function of voltages at that bus and others directly connected to it.

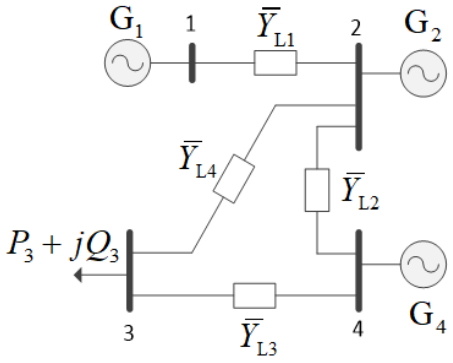


Figure 2.2: Network diagram for example to compute Y_{bus}

Let us consider a 4-bus network illustrated in Figure 2.2 where the buses are interconnected through transmission lines with the following admittances:

– $\bar{Y}_{L1}$: Line between Bus 1 and Bus 2

- $\bar{Y}_{L2}$: Line between Bus 2 and Bus 4
- $\bar{Y}_{L3}$: Line between Bus 3 and Bus 4
- $\bar{Y}_{L4}$: Line between Bus 2 and Bus 3

Applying Kirchhoff's Current Law (KCL) at each bus and using the admittance relationships, we obtain the following nodal equations:

$$\begin{aligned}
\bar{I}_1 &= \bar{Y}_{L1}\bar{U}_1 - \bar{Y}_{L1}\bar{U}_2 \\
\bar{I}_2 &= -\bar{Y}_{L1}\bar{U}_1 + (\bar{Y}_{L1} + \bar{Y}_{L2} + \bar{Y}_{L4})\bar{U}_2 - \bar{Y}_{L4}\bar{U}_3 - \bar{Y}_{L2}\bar{U}_4 \\
\bar{I}_3 &= -\bar{Y}_{L4}\bar{U}_2 + (\bar{Y}_{L3} + \bar{Y}_{L4})\bar{U}_3 - \bar{Y}_{L3}\bar{U}_4 \\
\bar{I}_4 &= -\bar{Y}_{L2}\bar{U}_2 - \bar{Y}_{L3}\bar{U}_3 + (\bar{Y}_{L2} + \bar{Y}_{L3})\bar{U}_4
\end{aligned}$$

The diagonal terms (e.g., $\bar{Y}_{L1}\bar{U}_1$) represent the total admittance connected to the bus. The off-diagonal terms (e.g., $-\bar{Y}_{L1}\bar{U}_2$) represent the influence of neighboring bus voltages through mutual admittance. We can now express these four equations as follows:

$$\begin{bmatrix} \bar{I}_1 \\ \bar{I}_2 \\ \bar{I}_3 \\ \bar{I}_4 \end{bmatrix} = \begin{bmatrix} \bar{Y}_{L1} & -\bar{Y}_{L1} & 0 & 0 \\ -\bar{Y}_{L1} & \bar{Y}_{L1} + \bar{Y}_{L2} + \bar{Y}_{L4} & -\bar{Y}_{L4} & -\bar{Y}_{L2} \\ 0 & -\bar{Y}_{L4} & \bar{Y}_{L3} + \bar{Y}_{L4} & -\bar{Y}_{L3} \\ 0 & -\bar{Y}_{L2} & -\bar{Y}_{L3} & \bar{Y}_{L2} + \bar{Y}_{L3} \end{bmatrix} \cdot \begin{bmatrix} \bar{U}_1 \\ \bar{U}_2 \\ \bar{U}_3 \\ \bar{U}_4 \end{bmatrix}$$

where

$$Y_{\text{bus}} = \begin{bmatrix} \bar{Y}_{L1} & -\bar{Y}_{L1} & 0 & 0 \\ -\bar{Y}_{L1} & \bar{Y}_{L1} + \bar{Y}_{L2} + \bar{Y}_{L4} & -\bar{Y}_{L4} & -\bar{Y}_{L2} \\ 0 & -\bar{Y}_{L4} & \bar{Y}_{L3} + \bar{Y}_{L4} & -\bar{Y}_{L3} \\ 0 & -\bar{Y}_{L2} & -\bar{Y}_{L3} & \bar{Y}_{L2} + \bar{Y}_{L3} \end{bmatrix}. \tag{2.2}$$

- **Incidence Matrix Method**
  A more scalable and systematic approach uses the incidence matrix, which captures the topological relationships between buses and branches. Combined with the admittances of each branch, the Y_{bus} matrix can be constructed efficiently, making this method ideal for computer-based implementations and large-scale networks. This method is suitable for automated generation of Y_{bus} in large-scale power systems. Unlike the manual inspection method, which is practical only for small networks, this method provides a systematic and scalable framework for constructing Y_{bus}. The bus admittance matrix can be obtained using the following relationship:

$$\mathbf{Y}_{\text{bus}} = \mathbf{A}_M^\top \cdot \mathbf{Y}_p \cdot \mathbf{A}_M \tag{2.3}$$

  Here, Y_p is a diagonal matrix containing the admittances of individual branches (lines) with order $N_B \times N_B$, where N_B is the total number of branches. The incidence matrix A_M, of size $N_B \times N$ (with N being the number of buses), encodes the connectivity and direction of branches relative to buses.

  The example illustrated in Figure 2.2, we demonstrate how to compute the bus admittance matrix Y_{bus} using the incidence matrix method. The primitive admittance matrix $\mathbf{Y}_p$ is:

$$\mathbf{Y}_p = \begin{bmatrix} \bar{Y}_{L1} & 0 & 0 & 0 \\ 0 & \bar{Y}_{L2} & 0 & 0 \\ 0 & 0 & \bar{Y}_{L3} & 0 \\ 0 & 0 & 0 & \bar{Y}_{L4} \end{bmatrix}$$

The incidence matrix A_M relates the branches to the buses. The matrix is of size $N_B \times N$ (branches × buses), where N_B is the number of branches (4 in this case) and N is the number of buses (4 in this case).

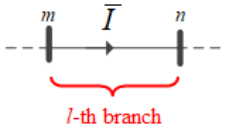


Figure 2.3: Elements of A_M

Each element a_{mn} depicted in Figure 2.3 in the incidence matrix represents the relationship between m and n and is expressed as follows:

- +1 if current l-th branch is directed away from node m.
- −1 if current l-th branch is directed towards m.
- 0 if l-th branch is not connected to m.

For this system, the incidence matrix A_M becomes:

$$\mathbf{A}_M = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ 1 & -1 & -1 & 0 \end{bmatrix}$$

Now, using Equation (2.3), we can compute the bus admittance matrix. The resulting matrix Y_{bus} will be:

$$\mathbf{Y}_{\text{bus}} = \begin{bmatrix} \bar{Y}_{L1} + \bar{Y}_{L4} & -\bar{Y}_{L1} - \bar{Y}_{L4} & -\bar{Y}_{L4} & 0 \\ -\bar{Y}_{L1} - \bar{Y}_{L4} & \bar{Y}_{L1} + \bar{Y}_{L2} + \bar{Y}_{L4} & \bar{Y}_{L4} & -\bar{Y}_{L2} \\ -\bar{Y}_{L4} & \bar{Y}_{L4} & \bar{Y}_{L3} + \bar{Y}_{L4} & -\bar{Y}_{L3} \\ 0 & -\bar{Y}_{L2} & -\bar{Y}_{L3} & \bar{Y}_{L2} + \bar{Y}_{L3} \end{bmatrix}$$

This matrix represents the admittance between each pair of buses, taking into account the connectivity and admittance of each branch. It is consistent with the result found using manual inspection. This process can be extended to larger networks, and the automated generation of Y_{bus} using the incidence matrix method becomes particularly useful for complex systems with many buses and branches.

An accurate Y_{bus} formulation is essential for solving various power system analysis problems, including power flow studies, fault analysis, and real-time state estimation.

2.1.2 Power Flow Equations

In power systems, the power flow analysis is an essential process for determining the voltages, currents, and power flows within the system. The power flow equations are based on a set of known inputs and help in understanding the steady-state behavior of the system.

Current and Voltage Relationships

To start, let's recall the relationship between the nodal current vector I and the nodal voltage vector U. This relationship is governed by the bus admittance matrix Y_{bus}, such that:

$$\mathbf{I} = \mathbf{Y}_{\text{bus}} \cdot \mathbf{U}$$

Here, I is $N \times 1$ vector of currents at each bus, U is $N \times 1$ vector of voltage magnitudes at each bus, and Y_{bus} is the bus admittance matrix, which describes the relationship between the voltage and current. N is the total number of buses (or nodes) in the network.

The elements of Y_{bus} between buses i and j, are defined as:

$$\bar{Y}_{ij} = Y_{ij}\angle\theta_{ij} = Y_{ij}\Big(\cos(\theta_{ij}) + j\sin(\theta_{ij})\Big) = G_{ij} + jB_{ij} \tag{2.4}$$

Similarly, the voltage at each bus i, denoted $\bar{U}_i$, can be expressed as:

$$\bar{U}_i = U_i\angle\delta_i = U_i\left(\cos(\delta_i) + j\sin(\delta_i)\right) \tag{2.5}$$

where: - U_i is the voltage magnitude at bus i, - δ_i is the phase angle of the voltage at i.

Current Injection at Each Node

The total current injected at node i can be obtained by summing the contributions of the currents due to the voltages at all other nodes in the system. Thus, the current injected at node i is given by:

$$\bar{I}_i = \sum_{n=1}^{N} \bar{Y}_{in}\bar{U}_n \tag{2.6}$$

Active and Reactive Power Injection

The active power P_i and reactive power Q_i injected at i can be derived from the complex power S_i, which is the product of the voltage and the complex conjugate of the injected current:

$$\bar{S}_i = P_i + jQ_i = \bar{U}_i \cdot \bar{I}_i^* \tag{2.7}$$

Breaking down the real and imaginary components of the complex power:
- The active power P_i is given by:

$$P_i = \sum_{n=1}^{N} U_i U_n Y_{in}\cos(\delta_i - \delta_n - \theta_{in}) \tag{2.8}$$

- The reactive power Q_i is given by:

$$Q_i = \sum_{n=1}^{N} U_i U_n Y_{ij} \sin(\delta_i - \delta_n - \theta_{in}) \tag{2.9}$$

The above equations are essential for solving the power flow problem.

2.1.3 Network Node Types

In a power system, each node (or bus) has a specific type based on the variables that are known or unknown for that node. For every node i, four quantities are defined:

- Active power: P_i
- Reactive power: Q_i
- Voltage phasor amplitude: U_i
- Voltage phasor angle: δ_i

The active and reactive powers are functions of the voltage amplitudes and angles at the nodes:

$$P_i = f_p(U_1, \ldots, U_i, \ldots, U_N, \delta_1, \ldots, \delta_i, \ldots, \delta_N)$$

$$Q_i = f_q(U_1, \ldots, U_i, \ldots, U_N, \delta_1, \ldots, \delta_i, \ldots, \delta_N)$$

The voltage phasor amplitude U_i and the voltage phasor angle δ_i are considered state variables. The state vector for all the nodes can be expressed as:

$$\mathbf{x} = \begin{bmatrix} \delta & U \end{bmatrix}^T$$

Thus, for each node, there are two equations, one for active power P_i and one for reactive power Q_i. However, without a reference node, there would be infinite possible solutions. The reference (slack) node is a node where both the voltage phasor amplitude and angle are known. This is typically the node with the largest generator. The state vector can be written as:

$$\mathbf{x} = \begin{bmatrix} \delta_2, \ldots, \delta_N, U_{N_g+2}, \ldots, U_N \end{bmatrix}^T$$

Where N_g represents the number of generator buses.

Node Types and Their Characteristics

This section outlines the different node types and summarizes their key roles and properties within the system. Their characteristics presented in Table 2.1 help define structural and functional behavior.

Slack Bus (Reference Node)

The slack bus is a special node in the system where both the voltage amplitude and angle are known. It serves as the reference point for the entire network. Typically, the node with the largest generator is chosen as the slack bus. The active and reactive power at the slack bus is calculated, but it is not an unknown variable.

PV Bus (Generation Node)

A PV bus represents a node where both the active power P_i and the voltage magnitude U_i are specified. The unknowns at this node are the voltage angle δ_i and the reactive power Q_i. These buses are typically associated with generator nodes where the output power and voltage magnitude are controlled.

PQ Bus (Load Node)

A PQ bus represents a load node where both the active power P_i and the reactive power Q_i are specified. The unknowns at this bus are the voltage magnitude U_i and the voltage angle δ_i. These buses are typically associated with load nodes in the system.

This classification is crucial for solving the power flow equations and ensures that the network's power flow can be calculated accurately. The following table summarizes the node type:

Table 2.1: Network node types and their associated variables

Node Type	Number of Nodes	Known Variables	Unknown Variables	Number of Equations per Bus Type
Slack Bus (Reference) (i = 1)	1	$U_1, \delta_1 = 0$	P_1, Q_1	0
PV Bus (Generation) (i = $2 \dots N_g + 1$)	N_g	P_i, U_i	δ_i, Q_i	N_g
PQ Bus (Load) (i = $N_g + 2 \dots N$)	$N - N_g - 1$	P_i, Q_i	δ_i, U_i	$2(N - N_g - 1)$

where N_g is the number of generator buses and N is the total number of network buses.

2.1.4 Newton-Raphson Method

The power flow equations, which are highly nonlinear, cannot be solved analytically due to their complexity. Therefore, numerical methods like the Newton-Raphson method are used to iteratively solve these equations. The Newton-Raphson method linearizes the power flow equations at each iteration, gradually improving the estimates of voltage magnitudes and angles at the network nodes. The method follows these key steps:

1. Iteration Update

$$\mathbf{J}(\mathbf{x}^k)\Delta\mathbf{x}^k = \mathbf{h}(\mathbf{x}^k) \tag{2.10}$$

2. State Variable Update

$$\mathbf{x}^{(k+1)} = \mathbf{x}^k + \Delta\mathbf{x}^k \tag{2.11}$$

where $\mathbf{x}^k = \left[\delta_2, \dots, \delta_N, U_{N_g+2}, \dots, U_N\right]^T$ is the state vector of voltage angles δ and magnitudes U at iteration k.

Mismatch Power Flow Equations

The mismatch equations are derived from the power flow equations and are expressed such that they become zero when the method converges. These mismatch equations are:

- Active Power Mismatch:

$$\Delta P_i(\mathbf{x}) = P_i - P_i(\mathbf{x}) = P_i - \sum_{n=1}^{N} U_i U_n Y_{in} \cos(\delta_i - \delta_n - \theta_{in}) \tag{2.12}$$

- Reactive Power Mismatch:

$$\Delta Q_i(\mathbf{x}) = Q_i - Q_i(\mathbf{x}) = Q_i - \sum_{n=1}^{N} U_i U_n Y_{in} \sin(\delta_i - \delta_n - \theta_{in}) \tag{2.13}$$

where P_i and Q_i are specified for node i, and the summation terms are iteratively computed for node i.

Jacobian Matrix

The Jacobian matrix J contains the partial derivatives of the power equations with respect to the state variables (voltage magnitude and angle) and is written as follows:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial P_2}{\partial \delta_2} & \cdots & \frac{\partial P_2}{\partial \delta_N} & \frac{\partial P_2}{\partial U_{N_g+2}} & \cdots & \frac{\partial P_2}{\partial U_N} \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \frac{\partial P_N}{\partial \delta_2} & \cdots & \frac{\partial P_N}{\partial \delta_N} & \frac{\partial P_N}{\partial U_{N_g+2}} & \cdots & \frac{\partial P_N}{\partial U_N} \\ \frac{\partial Q_{N_g+2}}{\partial \delta_2} & \cdots & \frac{\partial Q_{N_g+2}}{\partial \delta_N} & \frac{\partial Q_{N_g+2}}{\partial U_{N_g+2}} & \cdots & \frac{\partial Q_{N_g+2}}{\partial U_N} \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \frac{\partial Q_N}{\partial \delta_2} & \cdots & \frac{\partial Q_N}{\partial \delta_N} & \frac{\partial Q_N}{\partial U_{N_g+2}} & \cdots & \frac{\partial Q_N}{\partial U_N} \end{bmatrix}$$

The corrections matrix Δx and mismatch vector $h(x)$ solved at every iteration are given by:

$$\Delta \mathbf{x} = \begin{bmatrix} \Delta \delta_2^{(k)} \\ \vdots \\ \Delta \delta_N^{(k)} \\ \Delta U_{N_g+2}^{(k)} \\ \vdots \\ \Delta U_N^{(k)} \end{bmatrix} \qquad \mathbf{h}(\mathbf{x}) = \begin{bmatrix} \Delta P_2^{(k)} \\ \vdots \\ \Delta P_N^{(k)} \\ \Delta Q_{N_g+2}^{(k)} \\ \vdots \\ \Delta Q_N^{(k)} \end{bmatrix}$$

The matrix is split into four submatrices:

$$\mathbf{J} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix}, \tag{2.14}$$

where each term inside the matrix is written as follows:

Block J_{11}:

$$\text{Diagonal terms:} \quad \frac{\partial P_i}{\partial \delta_i} = -\sum_{\substack{n=1 \\ n \neq i}}^{N} U_i U_n Y_{in} \sin(\theta_{in} + \delta_n - \delta_i)$$

$$\text{Off-diagonal terms:} \quad \frac{\partial P_i}{\partial \delta_j} = U_i U_j Y_{ij} \sin(\theta_{ij} + \delta_j - \delta_i), \quad j \neq i$$

Block J_{12}:

$$\text{Diagonal terms:} \quad \frac{\partial P_i}{\partial U_i} = 2U_i Y_{ii}\cos(\theta_{ii}) + \sum_{\substack{n=1 \\ n\neq i}}^{N} U_n Y_{in}\cos(\theta_{in}+\delta_n-\delta_i)$$

$$\text{Off-diagonal terms:} \quad \frac{\partial P_i}{\partial U_j} = U_i Y_{ij}\cos(\theta_{ij}+\delta_j-\delta_i), \quad j\neq i$$

Block J_{21}:

$$\text{Diagonal terms:} \quad \frac{\partial Q_i}{\partial \delta_i} = \sum_{\substack{n=1 \\ n\neq i}}^{N} U_i U_n Y_{in}\cos(\theta_{in}+\delta_n-\delta_i)$$

$$\text{Off-diagonal terms:} \quad \frac{\partial Q_i}{\partial \delta_j} = -U_i U_j Y_{ij}\cos(\theta_{ij}+\delta_j-\delta_i), \quad j\neq i$$

Block J_{22}:

$$\text{Diagonal terms:} \quad \frac{\partial Q_i}{\partial U_i} = -2U_i Y_{ii}\sin(\theta_{ii}) - \sum_{\substack{n=1 \\ n\neq i}}^{N} U_n Y_{in}\sin(\theta_{in}+\delta_n-\delta_i)$$

$$\text{Off-diagonal terms:} \quad \frac{\partial Q_i}{\partial U_j} = -U_i Y_{ij}\sin(\theta_{ij}+\delta_j-\delta_i), \quad j\neq i.$$

2.1.5 Iterative Power Flow Calculation

The Newton-Raphson method solves the power flow problem through a series of iterations. The key steps of this iterative process are outlined as follows:

1. **Build the admittance matrix:** Compute the network admittance matrix Y_{bus} based on the topology and line parameters of the power system.
2. **Initial estimation:** Assign initial guesses to the state variables, voltage magnitudes and angles.
3. **Calculate mismatches:** Use the current values of voltage magnitude and angle to compute the active and reactive power mismatches ΔP_i and ΔQ_i for each bus. These are assembled into the mismatch vector $h(x)$.
4. **Perform the stop test:** If the norm of the mismatch vector $h(x^{(k)})$ is smaller than a predefined tolerance (e.g., $\varepsilon = 10^{-4}$), the algorithm stops and the solution is considered converged. Otherwise, go to next step.
5. **Construct the Jacobian matrix:** Form the Jacobian matrix J containing the partial derivatives of the power flow equations with respect to the state variables (voltage magnitudes and angles).
6. **Calculate corrections:** Solve the linear system to determine the update vector $\Delta\mathbf{x}^{(k)}$.
7. **Update the state:** Apply the corrections to the state vector: $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta\mathbf{x}^{(k)}$. Increment k and return to Step 3.

2.2 Approach to Study Steady-state Performance

Understanding the steady-state behavior of a power system is essential for ensuring reliable and secure operation under normal and changing conditions. A structured approach is outlined to evaluate steady-state performance through a sequence of modeling, simulation, and analysis steps. The process begins with the system layout and component parameters, which define the topology and electrical characteristics of the network elements such as buses, lines, transformers, and generators. These inputs form the foundation for constructing the power flow model, which mathematically represents the system using the Y_{bus} and nonlinear power flow equations. An iterative solution method, typically Newton-Raphson explained in earlier section, is applied to solve the power flow equations under defined operating conditions and possible topology changes. This step yields the steady-state voltages and power flows in the system. Once a valid solution is obtained, relevant variables are selected and compared against their admissible limits to assess system adequacy. Common performance indicators include bus voltage deviations and line/transformer loading percentages.

An understanding of the individual characteristics of each system component is crucial in interpreting steady-state results accurately. Moreover, it is essential that all models and analytical tools used in the study are validated against measured system responses to ensure credible and meaningful results.

Steady-State Voltage Performance

The key performance indicators in the steady-state analysis is the assessment of voltage levels and power flow across all network buses. One index used for this purpose is the Quasi-Stationary Voltage Index (QSVI), which quantifies the deviation of actual voltage magnitudes from their permissible bounds and is defined as:

$$\text{QSVI} = \min\left\{1, \max_{i=1,\ldots,N_{\text{bus}}}\left(\frac{|\Delta u_{\text{actual}-i}|}{|\Delta u_{\text{limit}-i}|}\right)\right\}, \tag{2.15}$$

where N_{bus} is total number of buses in the system, $\Delta u_{\text{actual}-i}$ is absolute deviation of the voltage magnitude at bus i from its nominal value (e.g., 220 kV) and $\Delta u_{\text{limit}-i}$ is the maximum allowed voltage deviation at bus i (e.g., ±10%). This index reflects how close any bus voltage is to its respective limit. A value of QSVI close to 1 indicates that at least one bus is operating near or at its voltage constraint. A value significantly less than 1 suggests that all buses are within acceptable voltage margins.

The Power Flow Index (PFI) is employed to assess the steady-state thermal loading of branches in each operating condition. It provides an indication of how close the apparent power flows on transmission lines are to their admissible limits. The PFI is defined as:

$$\text{PFI} = \min\left\{1, \max_{i=1,\ldots,N_{\text{branch}}}\left(\frac{S_{\text{actual}-i}}{S_{\text{limit}-i}}\right)\right\} \tag{2.16}$$

where $S_{\text{actual}-i}$ is the actual magnitude of the apparent power flow in branch i, and $S_{\text{limit}-i}$ denotes the corresponding maximum allowable value. A value of PFI greater than 1 signifies that at least one branch is experiencing overloading, pointing to possible violations in thermal constraints and the need for operational adjustments.

Exempel 2.1 (Modified IEEE 9-bus test system) Given the system seen in Figure 2.4, the model will be simulated MATPOWER which is an open-source MATLAB toolbox. It is composed of a set of M-files designed for modeling and analyzing power systems under steady-state conditions. It provides essential tools for performing power flow and optimal power flow calculations efficiently. The toolbox is widely used in academic and industrial studies for its flexibility and ease of integration. MATPOWER can be downloaded from: `http://www.pserc.cornell.edu/matpower/`.

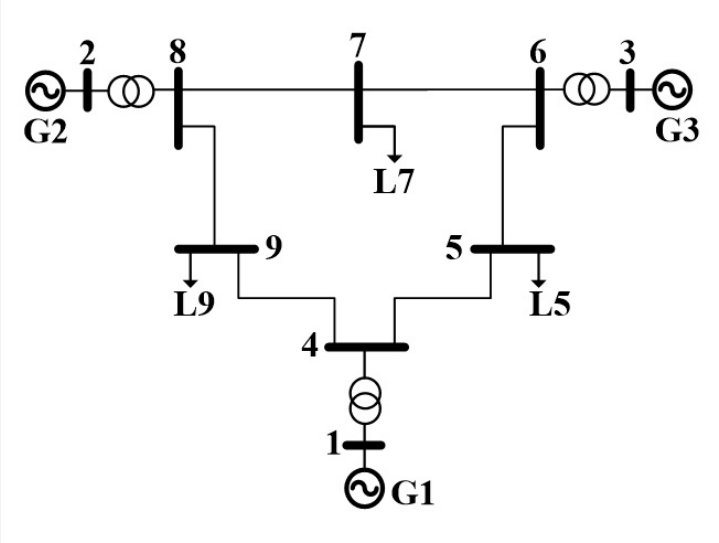


Figure 2.4: Example power-grid

The following Table 2.2 summarizes the known and unknown variables of the nodes:

Table 2.2: Network node types and their associated variables

Bus	Type	Known Variables	Unknown Variables
1	Slack	$U_{G1} = 1.04$ p.u. $\delta_{G1} = 0^o$	P_{G1} Q_{G1}
2	PV	$P_{G2} = 163$ MW $U_{G2} = 1.025$ p.u.	δ_{G2} Q_{G2}
3	PV	$P_{G3} = 85$ MW $U_{G3} = 1.025$ p.u.	δ_{G3} Q_{G3}
5	PQ	$P_{L5} = 90$ MW $Q_{L5} = 30$ Mvar	U_{G5} δ_{G5}
7	PQ	$P_{L7} = 100$ MW $Q_{L7} = 35$ Mvar	U_{G7} δ_{G7}
9	PQ	$P_{L9} = 125$ MW $Q_{L9} = 50$ Mvar	U_{G9} δ_{G9}

The data can be loaded as follows below.

```
%% Step 1: Load the data of the case
mpc_ieee9bus = loadcase('caseIEEE9bussys');
```

```
%% Step 2: Set the options for power flow calculation
mpopt = mpoption('pf.alg','NR', 'verbose', 3,
  'pf.tol', 1e-8, 'pf.enforce_q_lims', 2);

%% Step 3: Run power flow calculation
PFresults = runpf(mpc_ieee9bus, mpopt);
```

```
================================================================================
|     Bus Data                                                                 |
================================================================================
 Bus      Voltage          Generation             Load
  #   Mag(pu) Ang(deg)    P (MW)   Q (MVAr)    P (MW)   Q (MVAr)
-----  ------- --------  --------  --------  --------  --------
    1  1.040    0.000*    71.64     27.05       -         -
    2  1.025    9.280    163.00      6.65       -         -
    3  1.025    4.665     85.00    -10.86       -         -
    4  1.026   -2.217       -         -         -         -
    5  1.013   -3.687       -         -       90.00     30.00
    6  1.032    1.967       -         -         -         -
    7  1.016    0.728       -         -      100.00     35.00
    8  1.026    3.720       -         -         -         -
    9  0.996   -3.989       -         -      125.00     50.00
                         --------  --------  --------  --------
            Total:       319.64    22.84     315.00    115.00

================================================================================
|     Branch Data                                                              |
================================================================================
 Brnch   From   To    From Bus Injection    To Bus Injection     Loss (I^2 * Z)
  #      Bus    Bus    P (MW)   Q (MVAr)    P (MW)   Q (MVAr)    P (MW)   Q (MVAr)
-----  -----  -----  --------  --------  --------  --------  --------  --------
    1      1      4    71.64     27.05    -71.64    -23.92     0.000     3.12
    2      4      5    30.70      1.03    -30.54    -16.54     0.166     0.90
    3      5      6   -59.46    -13.46     60.82    -18.07     1.354     5.90
    4      3      6    85.00    -10.86    -85.00     14.96     0.000     4.10
    5      6      7    24.18      3.12    -24.10    -24.30     0.088     0.75
    6      7      8   -75.90    -10.70     76.38     -0.80     0.475     4.03
    7      8      2  -163.00      9.18    163.00      6.65     0.000    15.83
    8      8      9    86.62     -8.38    -84.32    -11.31     2.300    11.57
    9      9      4   -40.68    -38.69     40.94     22.89     0.258     2.19
                                                             --------  --------
                                                    Total:    4.641    48.38
```

Figure 2.5: Bus data and branch data results of the MATLAB command window

You can access information about available options by entering `help mpoption` in the MATLAB command window. The summary of results in the command window is presented in **??**.

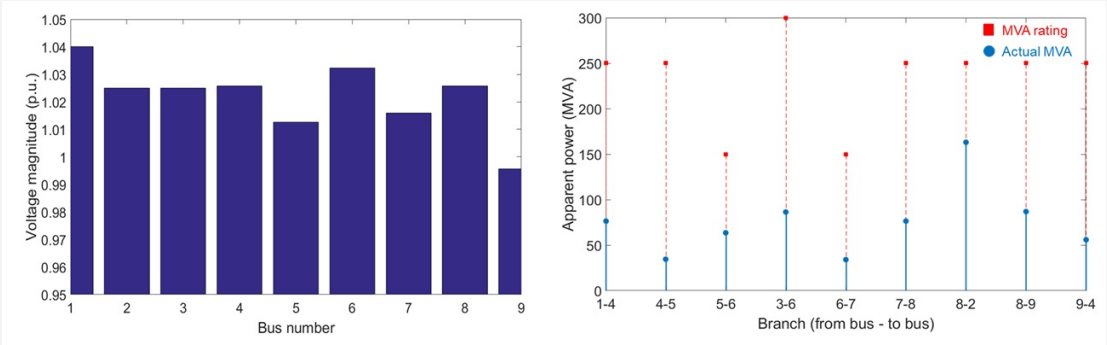
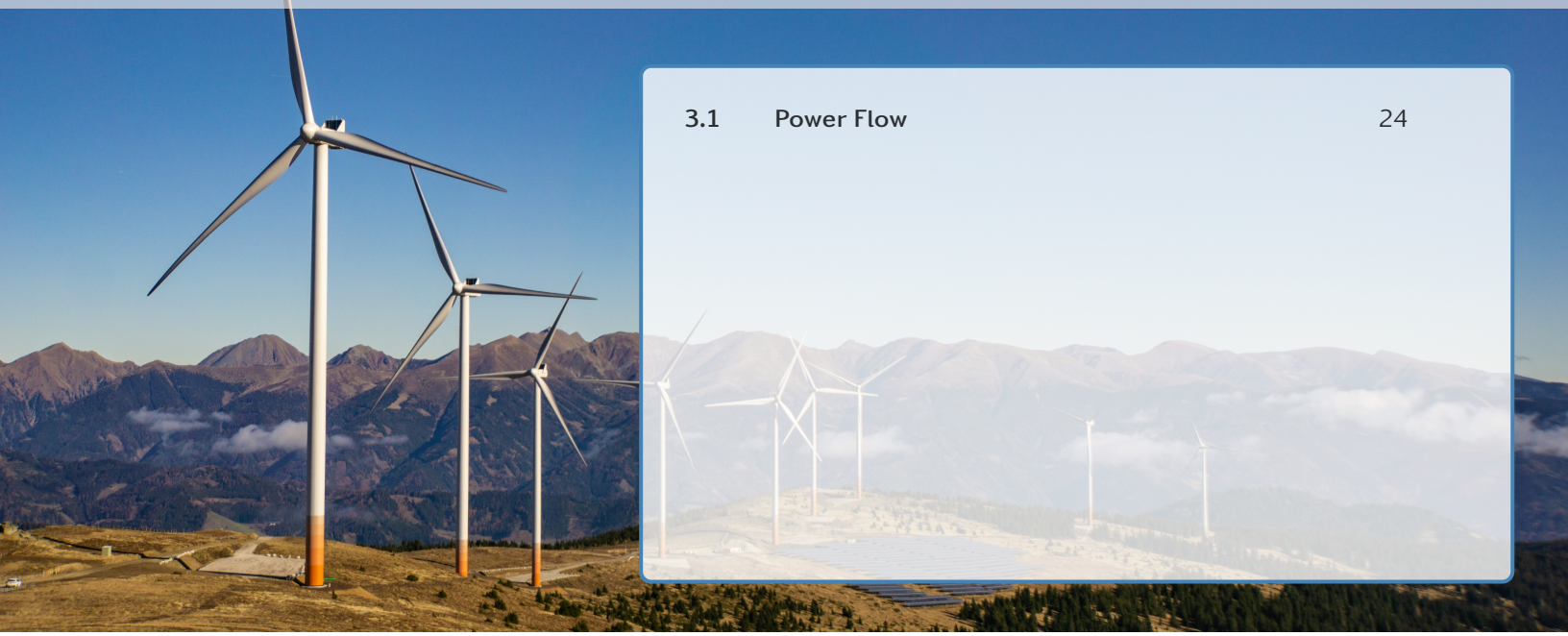


Figure 2.6: Bus voltages and line loadings results of the Pandapower analysis

As illustrated in Figure 2.6, the voltage magnitudes at nearly all buses exceed 1 per unit, indicating a relatively elevated voltage profile throughout the network. Regarding the branch loading, the apparent power ratings are predominantly around 250 MVA, whereas the actual power flows remain below 100 MVA in most cases. This suggests that the system operates well within its thermal limits, providing a significant margin for potential load growth or contingency handling.

3. Optimal Power Flow-based Techno-economic Assessment



This chapter concerns with the fundamentals of power flow calculations and the basic steps to implement them in computer-aided analysis of power system steady-state performance.

The learning objectives are:

(i) Formulate optimal power flow (OPF) for power system analysis.

(ii) Implement and solve OPF in a steady-state simulation package.

3.1 Power Flow

Generally speaking, the problem of power flow calculation is based on known inputs:

- Active power output of generators (except the slack generator) and demand.
- Reactive power of demand.
- Set-points of voltage magnitudes of generators with known active power output.
- Voltage magnitude and angle of the slack generator.

The fundamental difference between power flow calculation and OPF is illustratively shown in Figure 3.1. The OPF calculation finds the optimal values of certain (decision)

variables to achieve a desired operational target (e.g. minimum overall generation cost). The considered time horizon for operational planning can be long-term (e.g. several years), mid-term (e.g. months to years), short-term (e.g. days to months), day ahead (e.g.anticipating every hour of the next day), intra-day (e.g. every hour), intra-hour (e.g. every 15 minutes), or real-time (e.g. every actual second). The computational complexity and burden depends on the modelling depth and size of the power system under study, as well as on the number of objectives, the number of considered technical constraints, and the number of decision variables.

The modelling of the power system can consist for instance, of steady-sate (power flow) equations, non-linear equations describing dynamic performance. Besides, the model evaluation can consider single/multiple operational scenarios (e.g. to represent different demand levels and the corresponding dispatches of generation, compensation, and storage), together with the probabilities of the scenarios and/or disturbances (e.g. component outages) of interest of study.

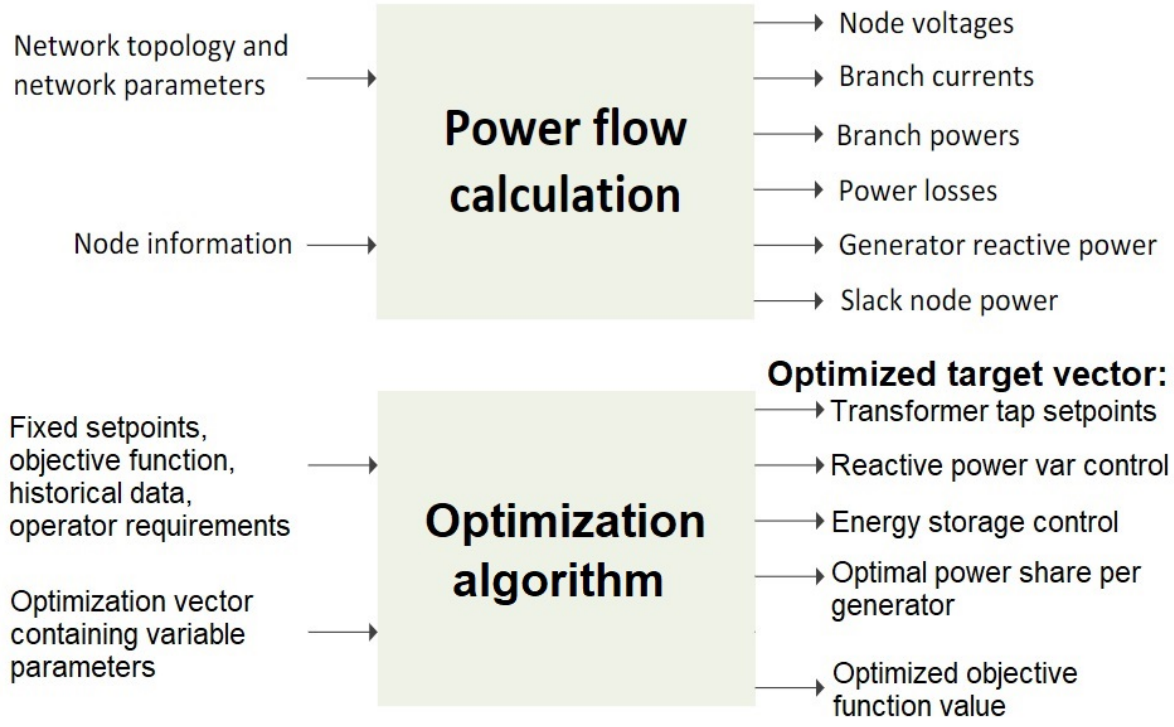


Figure 3.1: Schematic of steady-state power flow calculation and optimal power flow calculation.

3.1.1 Format of an Optimization Problem

Firstly, an Objective Function (OF) shall be defined, which could be a single function (i.e. single-objective problem) or a combinations of functions as seen in Equation 3.1 to be minimized or maximized [1]. For the generalized case of a multi-objective problem, it can be treated a summation of different Objective Functions $f_r(x)$, which can be related, for instance, to economic power dispatch parameters,environmental targets, reliability targets, to name a few examples. Each OF is evaluated by considering the actual values of an Opti-

mization Vector, consisting of a set of Optimization or Decision Variables, which constitute parameters that can be adjusted within predefined bounds of their associated controllability ranges (e.g. which are usually determined by operational experience, parametric sensitivity analysis, or feasible physical operational limits of devices).

$$OF = \sum_{r=1}^{\mathrm{p}} w_r \cdot f_r(\mathbf{x}) \tag{3.1}$$

In the approximation of a multi-objective problem by a single-objective problem, weighting factors w_r are assigned to the different Objective Functions in order to introduce bias so as to create a hierarchy according to the difficulty or importance of achieving particular objectives within a certain priority. Assigning the correct weights is a nuanced task. Hence, a sensitivity analysis is usually performed to defined them. Alternatively, the weights could also be defined as optimization variables.

Power system optimization problems are often subjected to a set of equality and/or inequality constraints. Such problems are denoted as constrained optimization. The constraints can concern with established operational limits (usually defined by operational experience and described in the so-called grid codes). For sake of illustration, the constraints are exemplified in Equation 3.2.

$$\begin{cases} g_i(\mathbf{x}) \leq 0, & i = 1, \ldots, m \\ h_j(\mathbf{x}) = 0, & j = m+1, \ldots, n \end{cases} \tag{3.2}$$

Besides, the minimum and maximum bounds of all optimization variables define the so-called the search space. The bounds are defined as Equation 3.3

$$x_k^{\min} \leq x_k \leq x_k^{\max}, \quad k = 1, \ldots, \mathrm{D} \tag{3.3}$$

There will be a small or large number of optimization variables depending on the scale of the problem. The computational complexity grows exponentially as the number of optimization variables increases. These form a Optimization or Solution Vector is displayed in Equation 3.4 [1].

$$\mathbf{x} = [x_1, x_2, \ldots, x_{\mathrm{D}}] \tag{3.4}$$

Several problem types are feasible: There may be a single-objective or multi-objective problem, with or without constraints, defined continuously or as a combinatorial of countable items, or a mix-integer problem.

Additionally, there may be certain complexities to it: The search, illustrated in Figure 3.2 space may be high-dimensional, the problem might be non-convex or with a discontinuous search space, it might be multimodal (it might be practically infeasible to find a global extremum, but rather only possible to define whether a local maximum or minimum satisfies the requirements of the system, i.e. the problem might have more than one solution), the problem might require too high numerical accuracy or might be highly non-linear, there might not be an analytical expression to describe the system, or the solution might incur a high computational burden.

InFigure 3.2, the search space will be defined by the minimum and maximum values of the optimization (objective) variables. Additionally, a set of constraints define a feasible space inside the search space.

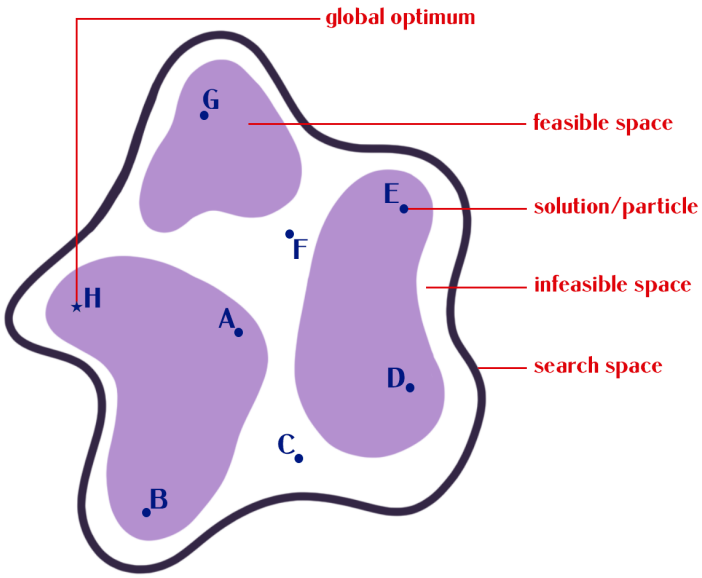


Figure 3.2: Search space of an optimization problem

The Solution Vector, which is iterated throughout the feasible space, can also be called "Particle". The goal is to achieve global optimality, although this is both hard-to-find and hard-to-prove.

3.1.2 Economic Power Dispatch Problems

In this example, the aim will be to minimize total dispatch cost of a power system, defined as in Equation 3.5. [1]

$$OF = \sum_{i=1}^{N_g} f_P^i\left(P_g^i\right) \tag{3.5}$$

Which is a summation of the Cost Function (of Power injected) for each generation unit that entails an economic remuneration. It can simply be a first order polynomial (constant factors), but it can be larger order polynomials or even a piece-wise combination of second order exponential polynomials plus sinusoidal terms, depending on how the relation between the profits and physical operation of generators and the costs from their operation are characterized (this may be to model wear and tear of components related to more complex control variables).

These cost functions may be evaluated by auditors once a year, for example, and they are subject to a number of constraints shown in Equation 3.6 [1], including nodal balance, which verifies that the power coming into a node equals the power going out of it by equating the power transfer from adjacent nodes plus the locally injected p_g, q_g and absorbed p_d, q_d power.

$$\begin{aligned} \mathbf{p}(\mathbf{v},\theta) - \mathbf{p}_\mathrm{g} + \mathbf{p}_\mathrm{d} &= \mathbf{0} \\ \mathbf{q}(\mathbf{v},\theta) - \mathbf{q}_\mathrm{g} + \mathbf{q}_\mathrm{d} &= \mathbf{0} \end{aligned} \tag{3.6}$$

As well as branch flow limits, seen in Equation 3.7 [1], which are simply related to the power transfer capacity ratings of the lines (usually related to thermal limits).

$$\mathbf{s} \leq \mathbf{s}_{\max} \tag{3.7}$$

The search space is then defined by a number of variables as per Equation 3.8, depending on the operational details of the generation units [1]. Normally, this includes at least the generation output powers, and for controllable units, in the case that energy storage is incorporated in the system, or other complexities of the system (e.g. controllable sources of reactive powers, phase shift changes, power electronic interfaced devices, etc.) become relevant.

$$\begin{aligned} &\mathbf{x}_{\min} \leq \mathbf{x} \leq \mathbf{x}_{\max} \\ &\mathbf{x} = \left[\theta_1, \ldots, \theta_{\mathrm{N_b}}, V_{\mathrm{m1}}, \ldots, V_{\mathrm{mN_b}}, P_{g1}, \ldots, P_{\mathrm{gN_g}}, Q_{g1}, \ldots, Q_{\mathrm{gN_g}}\right] \end{aligned} \tag{3.8}$$

There might be certain complexities to these problems. For example, how is the starting point of the optimization defined? Often the initial solutions are based on operational experience. What can be done if a new system is being designed? It is possible to, for example, generate expected values inspired from operational experience of real systems. Another approach is to create an initialization algorithm that finds initial conditions satisfying the constraints before the optimization is formally performed.

3.1.3 Reactive Power Dispatch Problems

The goal will be to minimize total losses in the system. In order to do so, the losses in every branch will be added (with number of branches N_k) of the system. The losses are functions of the voltage magnitudes and angles, both which may be constrained. The power loss function is defined as in Equation 3.9. [7]

$$P_{\mathrm{loss}} = \sum_{k \in \mathrm{N_K}} \mathrm{g}_k \left(V_i^2 + V_j^2 - 2V_i V_j \cos\theta_{ij}\right) \tag{3.9}$$

For the constraints, there is a nodal power balance and branch flow limit, but this time there is also a bus voltage magnitude constraint, which should be feasible within pre-defined lower and upper technical bounds. This can be seen in Equation 3.10.

$$\begin{aligned} &\mathbf{p}(\mathbf{v}, \boldsymbol{\theta}) - \mathbf{p}_{\mathrm{g}} + \mathbf{p}_{\mathrm{d}} = \mathbf{0} \\ &\mathbf{q}(\mathbf{v}, \boldsymbol{\theta}) - \mathbf{q}_{\mathrm{g}} + \mathbf{q}_{\mathrm{d}} = \mathbf{0} \\ &\mathbf{v}_{\min} \leq \mathbf{v} \leq \mathbf{v}_{\max} \\ &\mathbf{s} \leq \mathbf{s}_{\max} \end{aligned} \tag{3.10}$$

The Optimization Vector in this case is different from the one from Economic Dispatch Optimizations, as it tries to adjust all controllable parameters that affect the voltage profile of the system's nodes, including the reactive power of generators, controllable compensation devices (i.e. capacitor banks), transformers with on-load tap changers (a mechanism that adjusts the number of windings to adjust the voltage), etc. See Equation 3.11 [7].

$$\begin{aligned} &\mathbf{X}_{\min} \leq \mathbf{X} \leq \mathbf{X}_{\max} \\ &\mathbf{x} = \left[Q_{\mathrm{g1}}, \ldots, Q_{\mathrm{gN_g}}, Q_{\mathrm{c1}}, \ldots, Q_{\mathrm{cN_c}}, \mathrm{tap}_1, \ldots, tap_{\mathrm{N_{oLTc}}}\right] \end{aligned} \tag{3.11}$$

3.1.4 Solution Approaches

Solvers may either be conventional or heuristic, as seen in Figure 3.3.

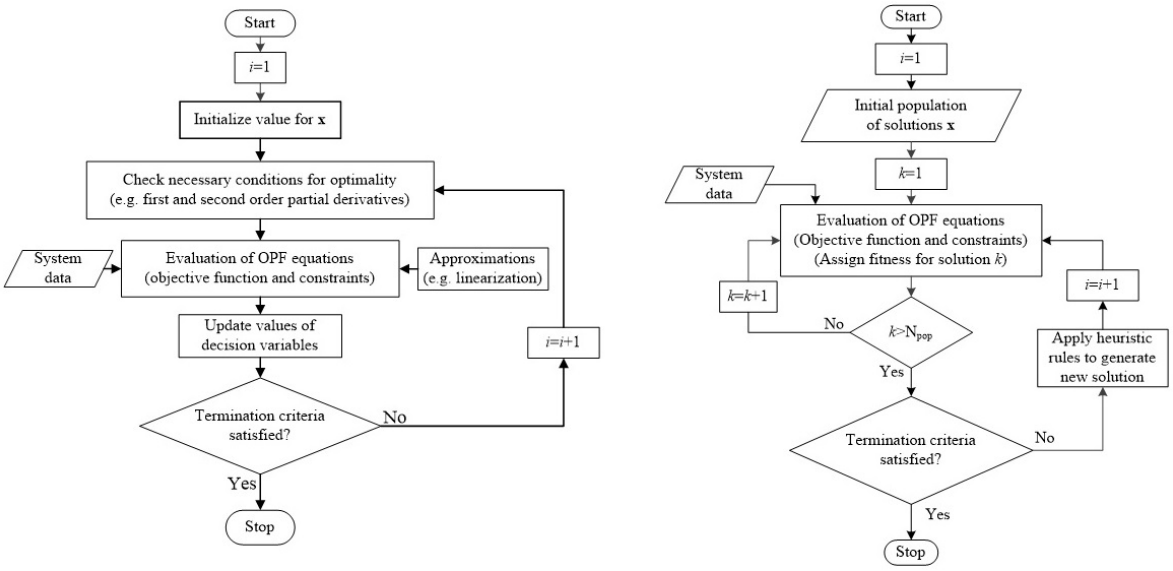


Figure 3.3: Flowchart for Conventional (left) and Heuristic (right) solvers

Conventional solvers start with one solution from previous experience or expected feasibility, then, they iterate while checking whether the conditions of optimality defined by the problem are satisfied by for example evaluating the power flow equations. The initial vector's values will then be updated by parameters defined in the iteration algorithm (Newton Raphson, sequential, quadratic, etc.).

In heuristic solvers, a set of initial solutions is considered, and the degree of violation of constraints is calculated in each iteration, and possible penalization factors that can be defined from them. These constitute the fitness of the solution, which is used as the optimality indicator for the termination criteria. Heuristic solvers can generate a set of solutions, unlike conventional solvers which usually work by iterating a single solution.

3.1.5 Be Careful with Stochastic/Learning Solvers

Stochastic or learning-based solvers, such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Differential Evolution (DE) can explore complex solution spaces but require careful tuning and evaluation. These solvers introduce randomness in solution generation and evolution, often inspired by biological or physical processes. The flowcharts for these solvers are illustrated in Figure 3.4.

Key concerns include:

- Randomness: These algorithms use pseudo-random numbers, which may lead to reproducibility issues unless properly seeded.
- Parameter Sensitivity: Algorithm performance can be highly dependent on chosen parameters such as mutation rate (GA), inertia weight (PSO), or differential weight (DE).
- Stochastic Outcomes: Results vary across runs, requiring statistical evaluation.

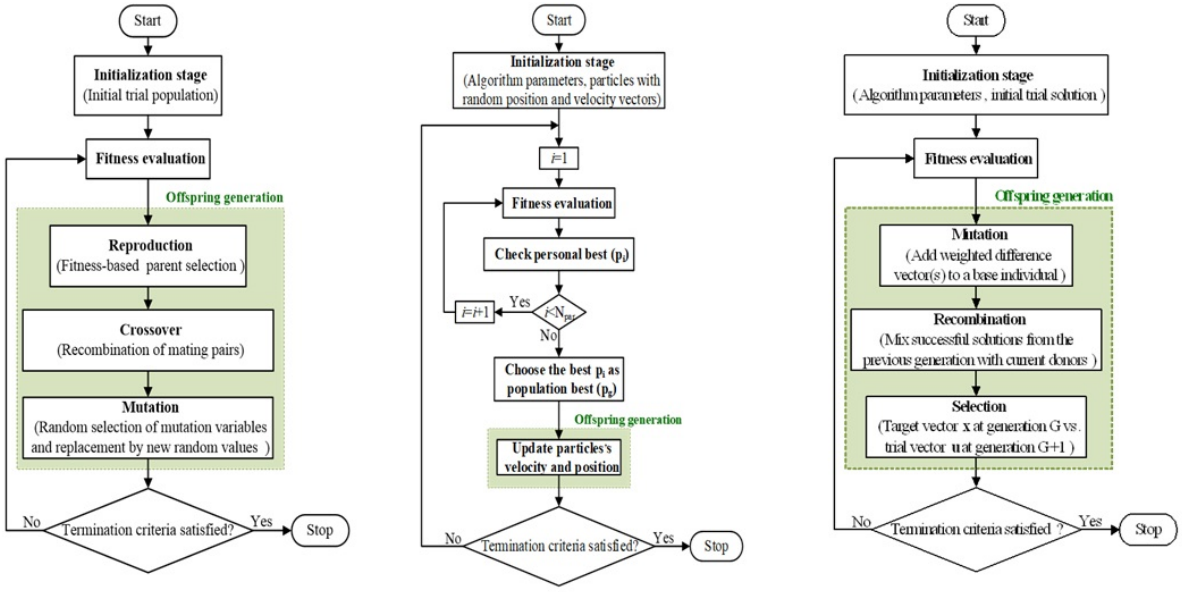


Figure 3.4: Flowchart for Genetic Algorithm (left), Particle Swarm Optimization (middle) and Differential Evolution solvers

Most random numbers are not truly random. Pseudo-random numbers are generated via deterministic algorithms. For example, in MATLAB:

Exempel 3.1 `rand, rand, rand, rand`

```
% Output: 0.8147, 0.9058, 0.1270, 0.9134
```

Restarting MATLAB yields the same output unless the seed is changed:

```
seed = sum(100*clock);
rng(seed);
% Alternatively: rand('state', sum(100*clock))
```

The selection of an optimization algorithm also depends significantly on the time horizon of the problem

- Short-term problems: Require reduced search time (seconds to a few hours).
- Mid- and long-term problems: Search time may extend to several hours or even days, allowing more complex algorithmic strategies.

When comparing optimization methods, several evaluation criteria are essential:

- Solution quality: How close is the result to the true optimum?
- Robustness: How consistent are the results across multiple runs?
- Convergence speed: How quickly does the solver reach an acceptable solution?
- Computational effort: What are the memory and time requirements?
- Tradeoff: How does one balance search time against the quality of the final solution?

Optimization outcomes are influenced by random variables. Thus, a single run does not constitute evidence of superiority. Monte Carlo Simulation (MCS) is used to repetitively execute the algorithm. A tradeoff exists between statistical confidence and computational effort. A possible stopping criterion is:

$$\frac{\sigma}{\mu} < \epsilon, \tag{3.12}$$

where σ is standard deviation, μ is mean value and ϵ is pre-specified error.

To quantitatively assess the performance of stochastic optimization algorithms, a variety of statistical metrics can be employed. Given a set of collected fitness values:

$$f = [f_1, f_2, \ldots, f_{N_{MCS}}]$$

the mean value is given by:

$$\mu = \frac{1}{N_{MCS}} \sum_{i=1}^{N_{MCS}} f_k, \tag{3.13}$$

where N_{MCS} is the number of Monte Carlo simulations. The standard deviation is:

$$\sigma = \sqrt{\frac{1}{N_{MCS} - 1} \sum_{i=1}^{N_{MCS}} (f_k - \mu)^2} \tag{3.14}$$

the best-case performance:

$$f_{best} = \min_{k=1\ldots N_{\mathrm{MCS}}} f_k$$

whereas the worst-case performance:

$$f_{worst} = \max_{k=1\ldots N_{\mathrm{MCS}}} f_k$$

Box plots summarize the distribution of collected fitness values through five key statistics: the smallest result, the lower quartile (value greater than 25% of all results), the median (value greater than 50% of all results), the upper quartile (value greater than 75% of all results), and the largest result. The success rate (SR) is defined as the percentage of successful simulations among a total of N_{MCS} independent runs and is computed with respect to

- Known optimal values: How often does the solver reach or approximate a known minimum/maximum?
- Constraint satisfaction: How many feasible solutions are found across N_{MCS} runs?

Several important limitations must be considered in comparing algorithms:

- The choice of tolerance ϵ is arbitrary.
- Function evaluation count can skew fairness.
- A large number of N_{MCS} repetitions might be necessary for statistically meaningful results.

Exempel 3.2 (Modified IEEE 9-bus test system cont.) Given the system seen in Figure 3.5.

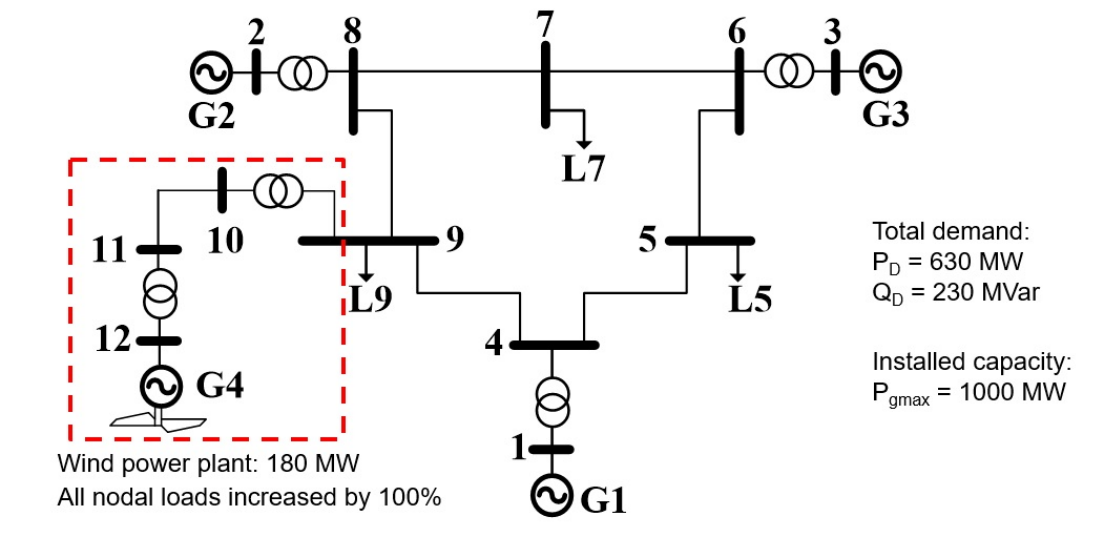


Figure 3.5: Example power-grid

Open the case in PandaPower and modify the parameters of the file by following the next steps.

```
## Step 1: Definition of system data
Open ieee9-wind.xlsx to check/modify data (e.g. parameters of buses,
loads, etc.) in the corresponding excel sheet.

## Step 2: Location of the wind power plant
Go to the excel sheet named as \trafo", check/change cell D5, which
defines the bus of interconnection of the wind power plant with system.

## Step 3: dispatch of wind power plant
Go to the excel sheet named as \sgen",
check/change cell D2, which defines Pout of the wind power plant,
check/change cells M2 and N2, which define min-max bounds of Pout,
check/change cell E2, which defines Qout of the wind power plant,
check/change cells G2 and H2, which define min-max bounds of Qout.
```

In PandaPower, to set a fixed setpoint of power for a generator, the maximum and minimum power need to be set to the same value. Otherwise, PandaPower will try to optimize the setpoint of the generator within the specified range. Since PandaPower is designed for synchronous generators, for this assignment, it is necessary to do this to simulate a renewable generator at a given available power. The same must be done for the reactive power setpoints.
It is possible to inject (positive) or absorb (negative) reactive power to tune the power factor.

Once renewables are dispatched at the available powers, the conventional generators can be dispatched as necessary to match demand.
In this step, the powerflow solution for the first iteration is initialized.

```
## Step 4: Set the options for opf and execute opf
Open opf_ieee9_wind.py
Line 5: Import system data from ieee9-wind.xlsx
Line 7: pp.runopp(net, init='pf', verbose=True)

#Info about opf
help(pp.runopp)
```

The parameter "verbose=True" prints the detailed results in PandaPower format. After running the code, the results revealed that the voltage profile is quite close to 1 per unit for all voltages as seen in Figure 3.6. The branch loadings are also mostly below 50%, except for the most loaded lines as depicted in Figure 3.7.

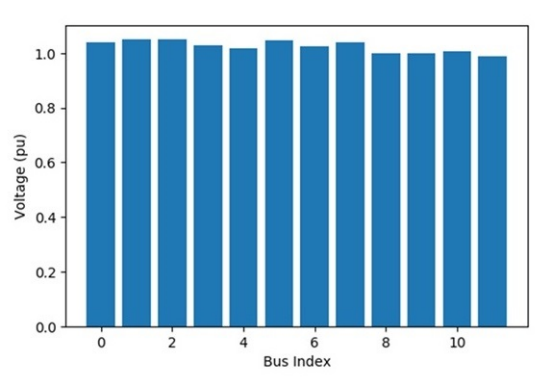


Figure 3.6: Bus voltage magnitude results of the Pandapower analysis

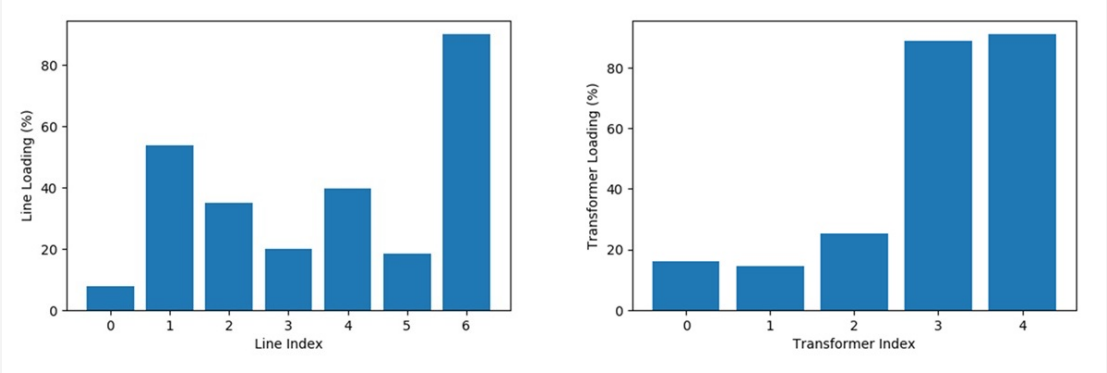
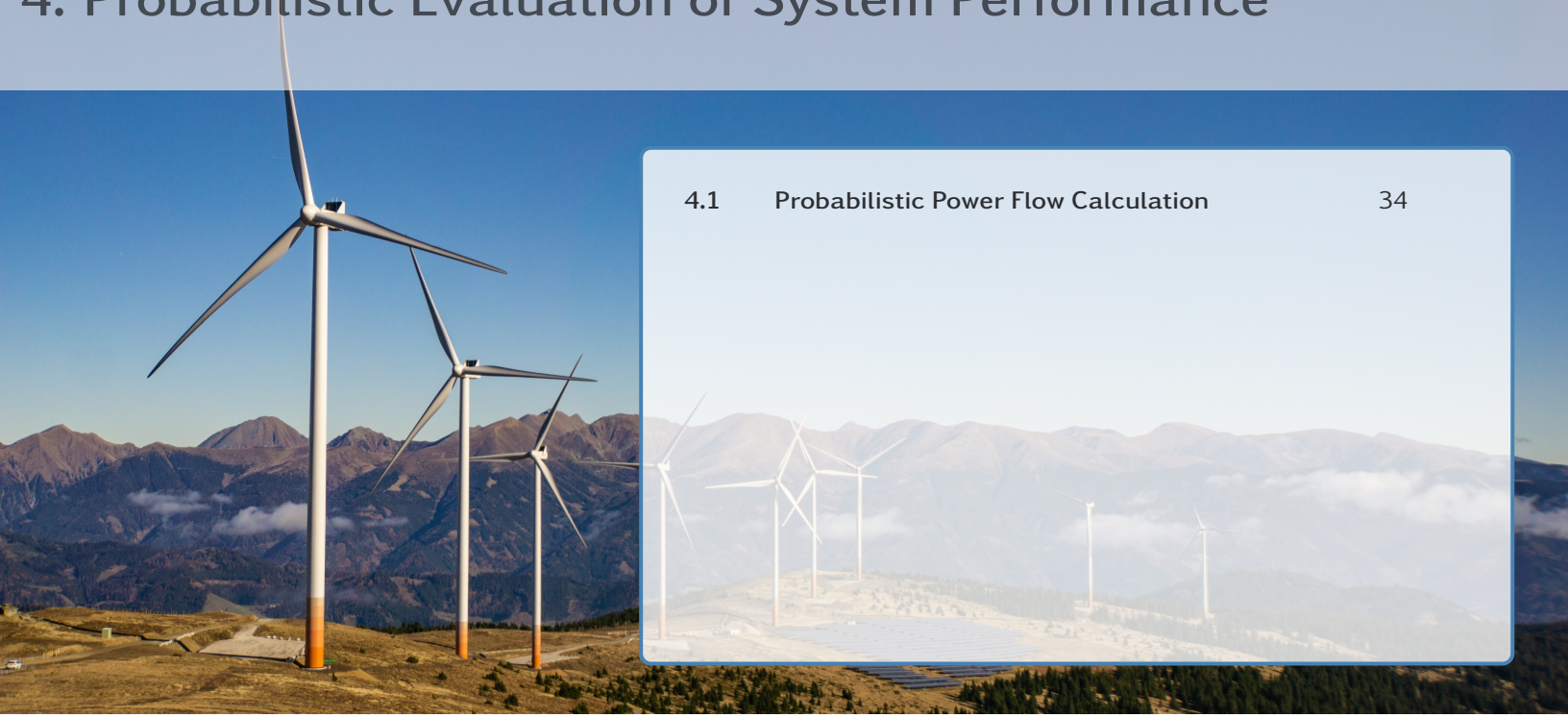


Figure 3.7: Branch loading results of the Pandapower analysis

4. Probabilistic Evaluation of System Performance



This chapter concerns with the fundamentals of power flow calculations and the basic steps to implement them in computer-aided analysis of power system steady-state performance.

The learning objectives are:

(i) Characterize uncertainties by using probabilistic distribution functions.

(ii) Implement Monte Carlo based simulations of probabilistic power flow.

(iii) Evaluate probabilistic steady-state performance of power systems.

4.1 Probabilistic Power Flow Calculation

4.1.1 Sources of uncertainty

When calculating power flows probabilistically, there can be many sources of uncertainty. The network's topology, parameters and settings may be too numerous or cumbersome to implement into a model (e.g. temperature-dependent line ratings), the load may vary in composition and parameters, both temporally and spatially, and the generation and storage systems may be subject varying parameters and patterns, such as weather.

The models themselves can have different topologies, compositions and parameters (i.e. a model could be of a steady snapshot of a system, or of a time-dependent dynamic system).

Stochastic models, exemplified in Figure 4.1 and 4.2, are generally used to account for uncertainties, making not a single prediction but a range of predictions. This type of model takes historical data from long-term measurements and fits it into a governing system of equations.

In the past, these models were most usually calibrated whenever a major event happens (often a system failure, in which case the practice is called post-mortem analysis), as these events tend to showcase the problems with predictive models. Nowadays, however, with more parameters than ever and comprehensive arrays of sensors and measurements, models tend to be updated and calibrated much more frequently.

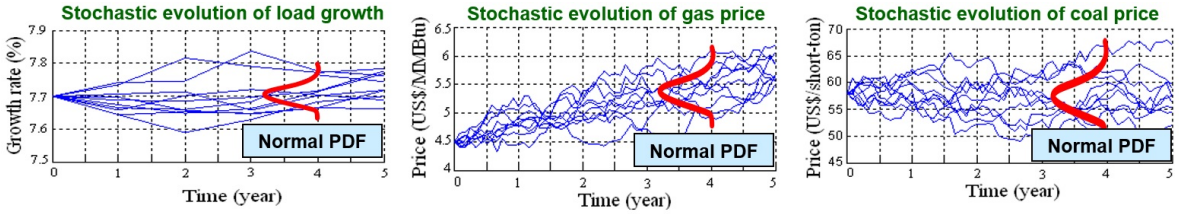


Figure 4.1: Mean-reverting stochastic process

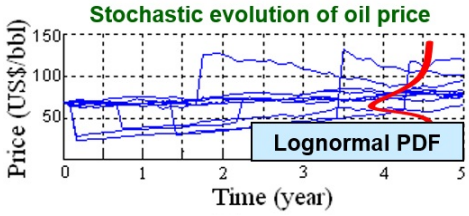


Figure 4.2: Poisson-Gaussian stochastic process. A lognormal distribution allows for better fitting of a variable with sudden jumps in value

As variables become more diverse, the traditional approach of approximating a single probabilistic distribution function (PDF) to a single normal has become obsolete, and models which take into account multiple PDF's into multiple normals have become a preferred option. The Gaussian Mixture Model is one of such models.

One challenge with these models is that they typically rely on an Expectation Maximization algorithm, which iteratively finds maximum probability estimates of the parameters in a stochastic model which depends on latent variables (variables that can't be observed but may be inferred). Expectation Maximization algorithms, however, are not great at minimizing the amount of individual components that are merged into the stochastic model.

4.1.2 Load modelling: Gaussian Mixture Model

The current trend in Load Modelling is to utilize Gaussian Mixture Models, which are parametric partial differential equations consisting on weighted sums of Gaussian Probabilistic Densities (N_c), as shown in Equation 4.1 [3].

$$p(\mathbf{x} \mid \lambda) = \sum_{i=1}^{N_C} w_i g(\mathbf{x} \mid \boldsymbol{\mu}_i, \Sigma_i) \tag{4.1}$$

Each D-variate component density in Equation 4.1 is defined in the manner seen in Equation 4.2, and its mixture weights satisfy the constraint expressed in Equation 4.3 in order to have a well balanced model. [3]

$$g(\mathbf{x} \mid \boldsymbol{\mu}_i, \Sigma_i) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Sigma_i|^{\frac{1}{2}}} e^{\left\{-\frac{1}{2}(x-\mu_i)' \Sigma_i^2 (x-\mu_i)\right\}} \tag{4.2}$$

$$\sum_{i=1}^{N_C} w_i = 1 \tag{4.3}$$

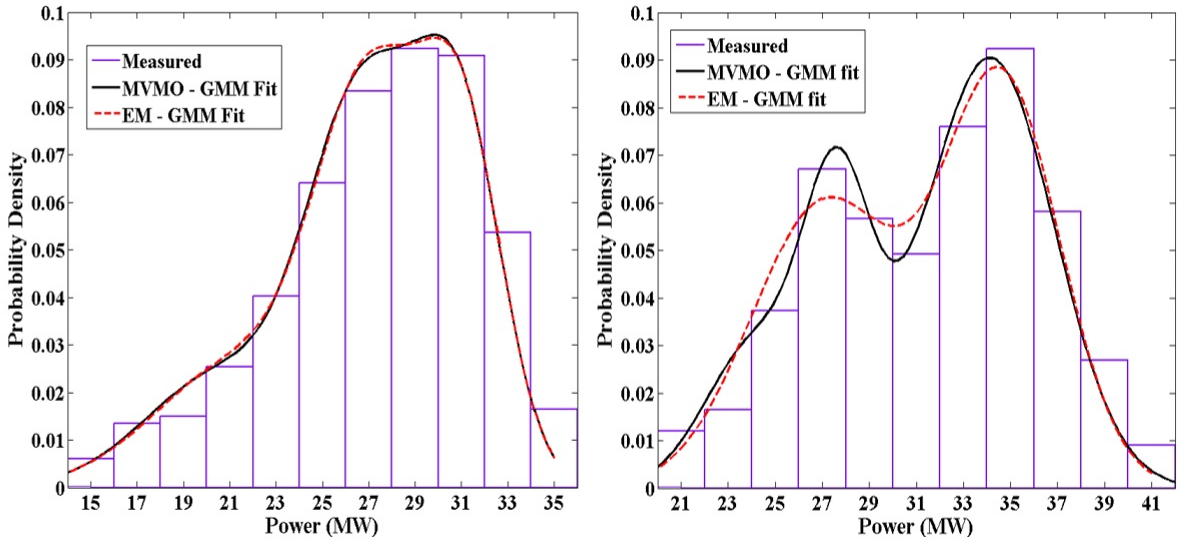


Figure 4.3: Gaussian Mixture Model approximation of the power probability density of two substations (Punto Fijo on the left, Judibana on the right) in the Venezuelan peninsula of Paraguaná. MVMO means Mean-variance mapping optimization algorithm. [3]

As can be seen in Figure 4.3, data that will not fit into a single Gaussian Distribution can be more accurately fit by a Gaussian Mixture Model. [3]

4.1.3 Wind speed modelling: Weibull Distribution

The Weibull distribution, shown in Figure 4.4, is a probabilistic model widely used to characterize the historical data of windspeed in a given location. It has a shape parameter that can be used to adjust the shape of the distribution, making it more bell-shaped, exponential-like or log-normal-like, as well as a scale parameter to scale it up or down.

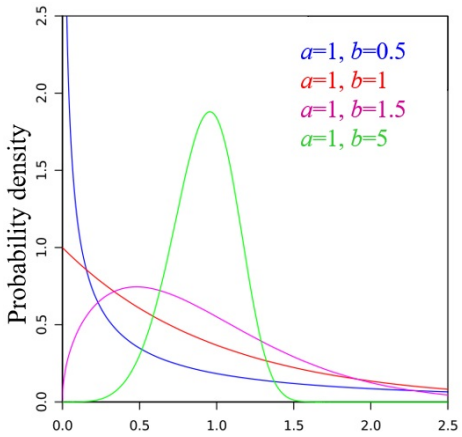


Figure 4.4: Weibull distributions for different parameter values. Figure used under the Attribution-ShareAlike 3.0 Unported license: https://creativecommons.org/licenses/by-sa/3.0/legalcode

To fit historical data to Weibull distributions Equation 4.4 is used, where a is the scale parameter and b is the shape parameter. [5]

$$f(x \mid a, b) = \frac{b}{a}\left(\frac{x}{a}\right)^{b-1} e^{-\left(\frac{x}{a}\right)^{b}} \tag{4.4}$$

4.1.4 Montecarlo simulations

A repetitive assessment is necessary to evaluate the probabilistic distribution functions generated by the Gaussian Mixture Models or Weibull Distributions. In a Monte Carlo Simulation, the performance of the system is evaluated by inputting the parameters from the probabilistic distribution functions and taking random samples of system parameters from them, roughly following the flowchart seen in Figure 4.5. The system is then evaluated for each sample (using the Power Flow equations, for example) and the optimal controllable parameters are then determined for these conditions (i.e. for optimal economic dispatch). The collection of the sampled and calculated parameters are used to analyzed the variable behavior of the system.

The simulation is stopped when the actually computed standard deviation of a certain random variable reflecting the performance of the system, as relatively compared with respect to the actually computed the expected value of that variable, reaches a certain steady behavior for a certain amount of Monte Carlo simulations.

Another stopping criterion consists of assuming that the mean value will follow an normal distribution function, and then the simulation can be stopped once the mean value reaches a certain probability within a normal distribution.

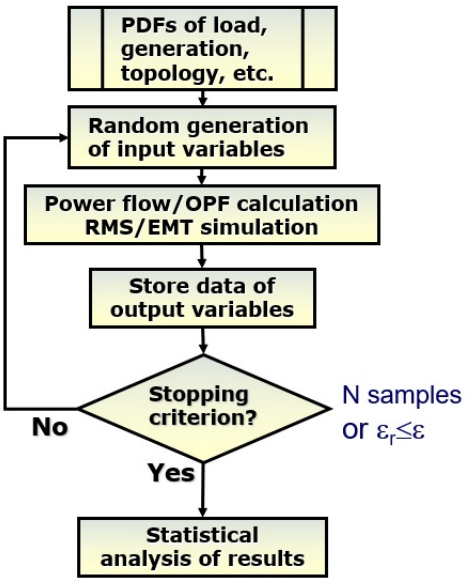


Figure 4.5: Flowchart of a Montecarlo Simulation

Hereby two example rules for stopping criteria based on mean error are used: Example Rule 1 seen in Equation 4.5 and Example Rule 2 seen in Equation 4.6. [6]

$$\varepsilon_r = \frac{\sigma\left(h_i(x)\right)_N}{E\left(h_i(x)\right)_N} \tag{4.5}$$

$$\varepsilon_r = \frac{\left[\Phi^{-1}(1-\delta/2)\right] \times \sqrt{\sigma\left(h_i(x)\right)_N / N}}{E\left(h_i(x)\right)_N} \tag{4.6}$$

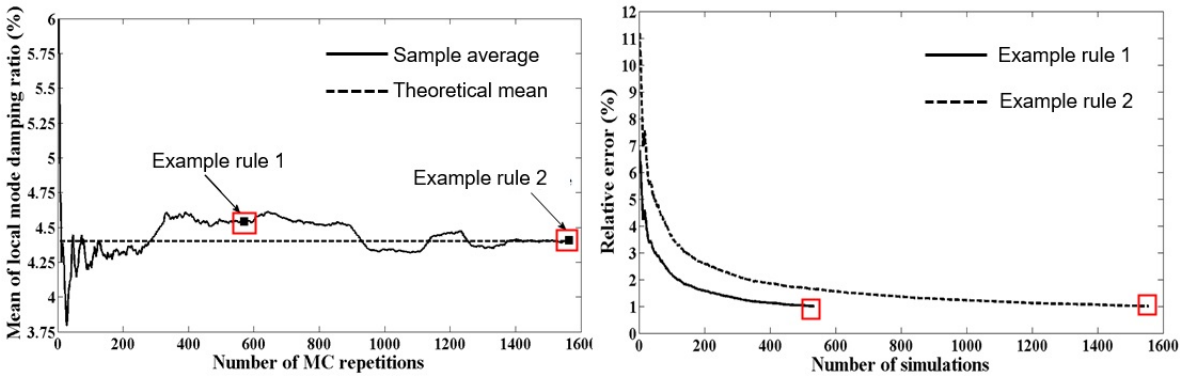


Figure 4.6: Montecarlo small signal stability analysis, showcasing Montecarlo and theoretical means vs. repetitions on the left and the relative error of the Montecarlo mean vs. the amount of simulations on the right. [6]

In Figure 4.6 [6], it can be seen that the Example Rule 1 does not ensure that the simulation actually converges, and therefore repetition of the process with differently seeded random parameters may yield different results. Example Rule 2, on the other hand, reaches a much higher degree of confidence. The degree of confidence is the main driving factor when defining the stopping criteria of Monte Carlo Simulations.

4.1.5 Probabilistic evaluation of power flow

All the information that can be approximated with probability distributions needs to be used in some way. It is used to fit a repetitive assessment, in order to evaluate the variable performance of the system.

To do a Monte Carlo-type analysis of the system, first the probabilistic functions are generated using historical data, and then using those functions an algorithm is utilised to generate a data set of random parameters that represent a set of probable states of the system, therefore simulating snapshots of a system in an asynchronous way that is consistent with each state's probability.

Then, optimization algorithms are used on each of those states to determine, for instance, the optimal economic dispatch, or reactive power dispatch, etc. This will generate a matrix of optimal parameter ranges for the performance variables, that will indicate, for instance which operational ranges could occur.

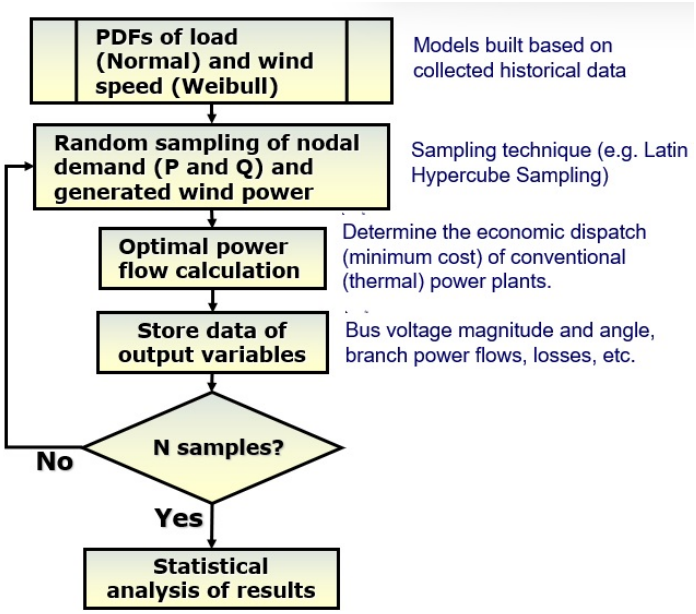


Figure 4.7: Flowchart of the probabilistic evaluation of powerflow

Exempel 4.1 (Modified IEEE 9-bus test system cont.) Before, the optimal power flow for a snapshot of a system state for the system shown in Figure **??** was found. Now the model will be simulated probabilistically, with a data-based approach.

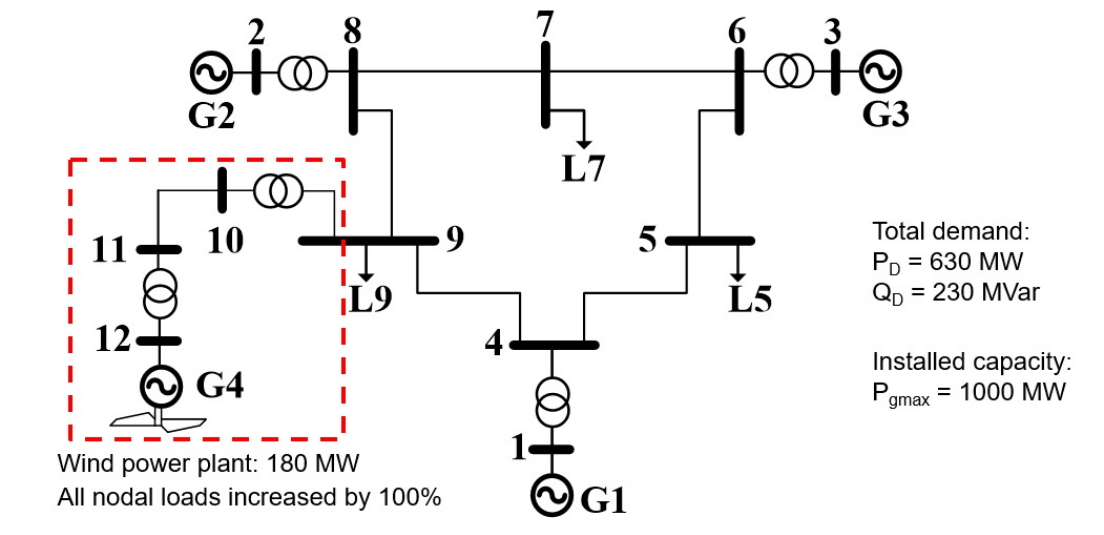


Figure 4.8: Example power-grid

First, create some normal functions for the demand, using an arbitrary degree of randomness. The PDF of the demand active power is as in Equation 4.7, where μ is taken as P value in bus data matrix and σ is generated with an uniform random distribution $[1, 30]$. The reactive power can be calculated from the active power and the power factor previously calculated for the nominal system (i.e. by assuming a constant power factor in any operating condition).

$$f(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{4.7}$$

For the wind speed, average speed data from the North Sea was used together with a shaping factor. The PDF of wind speed is exemplified in Equation 4.8, where example values are $a = 11$ and $b = 2.02$ and with the wind powerplant operating at unit power factor.

$$f(x \mid a, b) = \frac{b}{a}\left(\frac{x}{a}\right)^{b-1} e^{-\left(\frac{x}{a}\right)^{b}} \tag{4.8}$$

The general quadratic cost function of a thermal generator is displayed in Equation 4.9. It has a quadratic term for generator operational costs, a linear term for the power dispatch costs, and a constant term for fixed maintenance costs.

$$f_P\left(P_g\right) = C_2 P_g^2 + C_1 P_g + C_0 \tag{4.9}$$

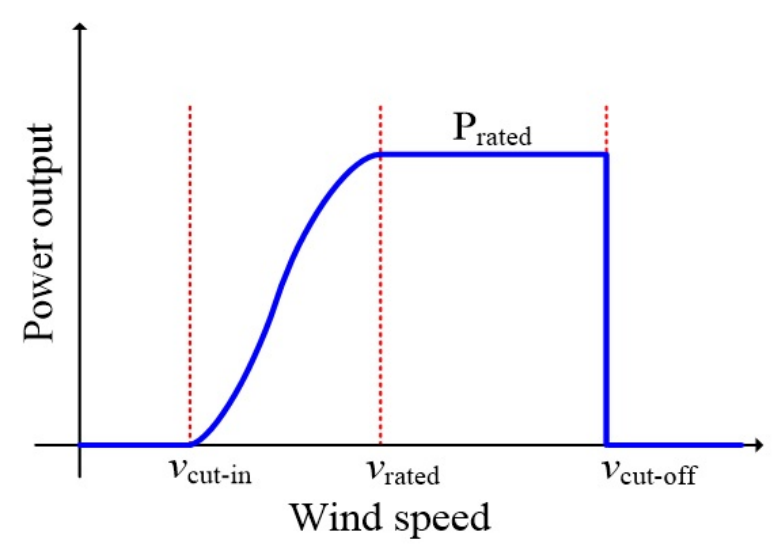


Figure 4.9: Power of a pitch-regulated wind generation system. [**power˙curve**]

The power curve of a pitch-regulated wind generation system looks as shown in Figure 4.9, and is described by Equation 4.10. Below cut-in wind speed, the turbine's blades are mostly stalled and not enough lift is produced to move the rotor. Between cut-in and rated wind speeds the turbine is in the pitch-regulated regime, where the transition up to nominal power is described by non-linear functions that can be approximated by polynomial functions, and is mostly dictated by the aerodynamic efficiency of the blades and how close to optimal is the pitch-control function used. Then at above rated wind speed the turbine will operate at nominal power, controlling its pitch to stay as precisely as possible at its rated power, until cut-off speed is reached, at which aerodynamic loads become too large for safe operation and the rotor is feathered (i.e. the blades' pitch is set to a zero-lift angle) and stopped.

$$\begin{cases} 0 & v \leq v_{\text{cut-in}} \\ \frac{\mathrm{P}_{\text{rated}}\left(v^3 - v_{\text{rated}}^3\right)}{\left(v_{\text{rated}}^3 - v_{\text{cut-in}}^3\right)} & v_{\text{cut-in}} < v < v_{\text{rated}} \\ \mathrm{P}_{\text{rated}} & v_{\text{rated}} \leq v < v_{\text{cut-off}} \\ 0 & v \geq v_{\text{cut-off}} \end{cases} \tag{4.10}$$

The reactive power injection has to be adjusted to match a voltage or power factor setpoint by means of VAR control (i.e. droop adjustment, tuning of power electronic converters' setpoints), illustrated in Figure 4.10, before the power can be injected to the grid. The voltage setpoint or power factor setpoint is selected by the grid operator and communicated to the wind farm operator as needed.

Fast control mechanisms kick in to react to large changes of the operational parameters of the system, and are mainly in place to prevent generator faults, and to collectively prevent grid outages during sudden (large gradient) changes.

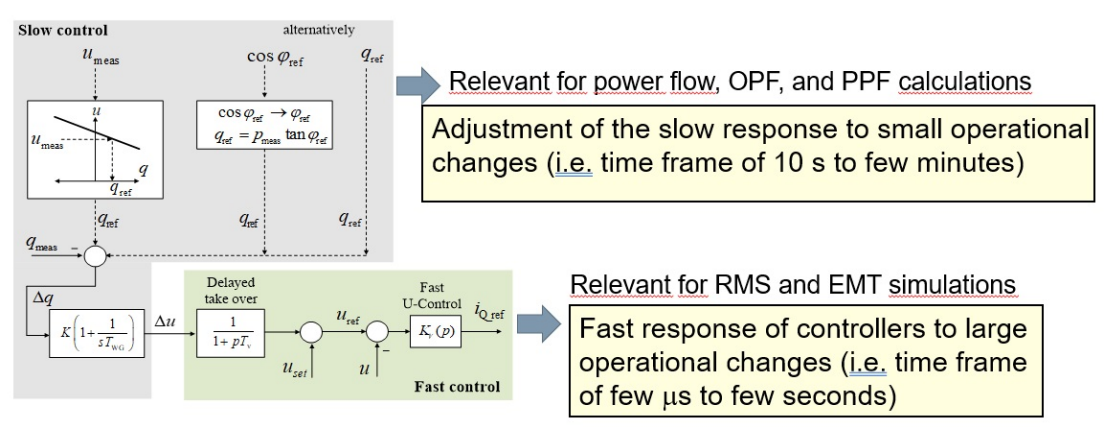


Figure 4.10: Var control scheme

The data can be loaded as follows below. First, define the parameters of the Weibull distribution used to map the windspeed probability density, as well as the possible loads and power factors.

```
## Step 1: Load the data of the case
Open the prob_opf_ieee9_wind.py in Spyder/Python IDLE
# Q/P ratio of loads
pq_phi_ratio_load= np.asarray((net.load.q_mvar/net.load.p_mw))

# Parameters of Weibull distribution of wind speed
k=2.02 #Shape parameter in pg,
lambda=11 #Scale parameter in m/s

#Parameters of wind power plant
Pwpp=180 #Max. MW of the wind power plant
wsin=3 # Cut in wind speed in (m/s)
wsr=12 #Rated wind speed in (m/s)
wsout=20 # Cut-off wind speed in (m/s)
wpcoshphi=1 #Operation at unit power factor
```

Then generate a database of randomly generated cases for analysis as follows.

```
## Step 2: Generate random variation wind power of nodal load demand
N=1000 #Amount of random samples to be generated
#Wind power
#Random wind speed following Weibull distribution
ws rand = np.asarray([random .weibullvariate(lambda_, k)
for _ in range (N)])
Pwpp_act = [
for i in ws_rand:
if i<wsin :
Pwpp_act.append(0)
```

```
elif i>wsin and i<wsr:
Pwpp_act.append(Pwpp^*(i^(**)3-wsin^^(**)3)/(wsr^(**)3-wsin**3))
elif i>wsr and i<w sout:
Pwpp_act.append(Pwpp)
elif i>w sout:
Pwpp_act.append(0)

## Step 3: Generate random variation wind power of nodal load demand

#Define mean values of nodal loads
load_mean= np.asarray(net.load.p_mw)

#Define standard deviations of nodal loads
load_std= np.random.uniform(1,30,3)

## Step 4: Run probabilistic power flow calculation
results= []
for wind_p in Pwpp_act:
#Create random load variations with normal distribution
net.load.p_mw = np.random.normal(means, sd)
net.load.q_mvar = np.asarray(net.lod.p_mw) * pq_ratio
#set wind powers
ner=t.sgen.p_mw = wind_p
<other>...
#run optimal power flow
try:
pp.runopp(net,init='pf',verbose=False)
#store results. More in python file
except:
continue
```

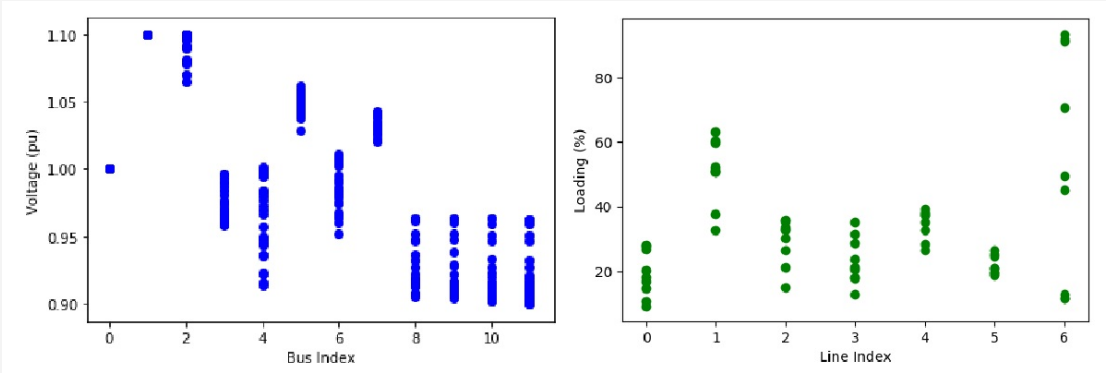
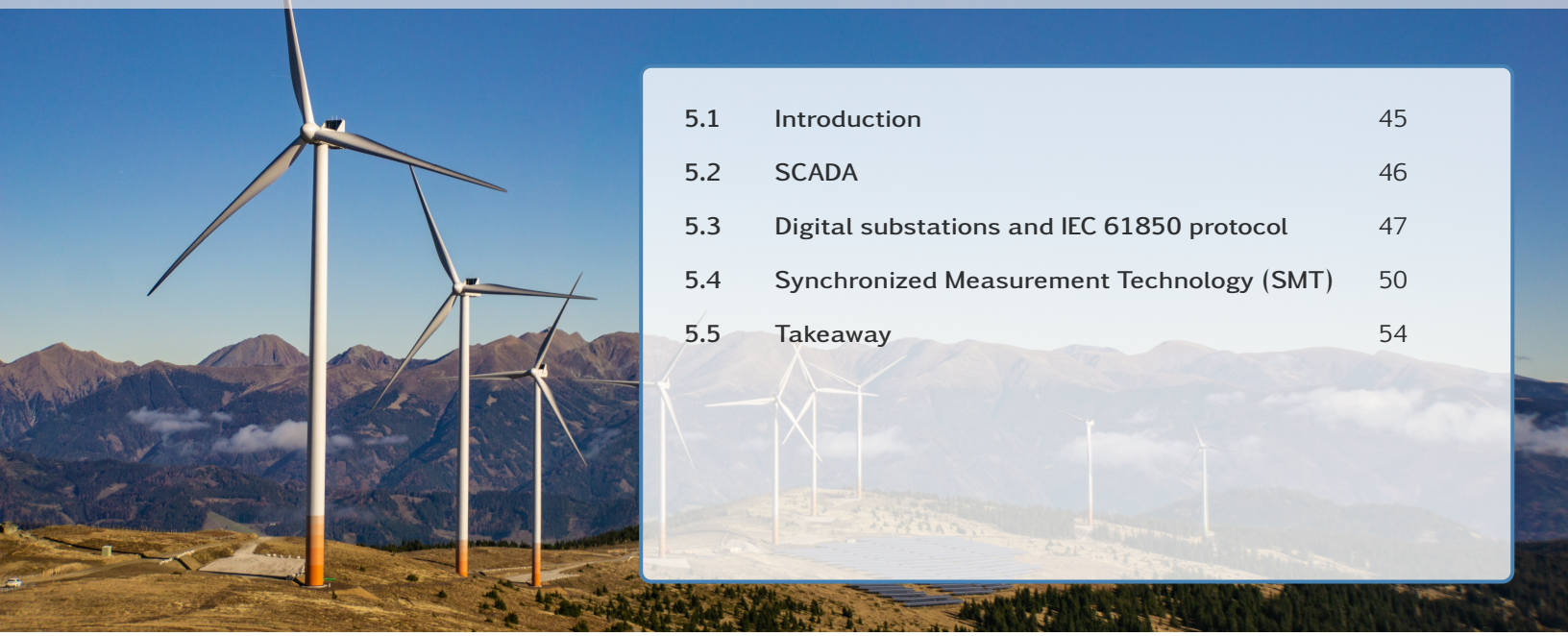


Figure 4.11: Bus voltages and line loadings results of the Pandapower analysis

In this case, as seen in Figure 4.11, there are under voltages in several buses (over 5% below nominal). This be improved by, for example, changing the power factor.

Besides, some of the line loadings are inching closer to 100%. Additional aspects (e.g. controllers) could be included in the model to enable the representation of measures to keep bus voltage magnitudes and line loading within desired values.

5. Energy Management and SCADA



This chapter covers energy management and SCADA.

5.1 Introduction

As seen in the previous chapters, the management of the electrical power system is a complex task, because the production and consumption of electrical power must be balanced all the time. When this balance is not met, the frequency of the system changes, leading to possible damage to equipment.

In the past, the traditional power system used to be hierarchically organized, with big production plants and uni-directional power flow. The loads were well-known and predictable with great accuracy. Nowadays, some loads, such as EV chargers, heat pumps and air conditioners, are becoming ”smarter” and interconnected in the IoT (Internet of Things). Therefore, they are becoming more and more unpredictable, because the models that regulate their functioning are more complex and inaccessible. Also generation is changing and becoming more dynamic, as renewable energy sources are intermittent and distributed, and have less inertia than traditional power plants. In this context, the power system has to be adapted to face these new challenges and to become more ”intelligent”. This can be achieved through monitoring and control systems, which will be introduced in this chapter.

The Energy Management System (EMS) is the system in control of the power grid. It optimally manages energy generation and transmission reliably and securely, to increase the reliability and efficiency of the transmission grid and meet the security and operation requirements. The original main task of the EMS is to reach the economically optimal dispatch of energy, taking into account the cost of generation and the constraints of the system. Additionally, the EMS is in charge of the monitoring and control of energy transmission and distribution and load forecasting and balancing.

The EMS is connected to other processes correlated to grid operation. Asset management is an important department, in charge of monitoring the aging of equipment and scheduling maintenance. The EMS exchanges data with asset management and vice versa: in this way the asset management department can elaborate the data to plan the maintenance, while the EMS knows when the equipment has to be put out of service for maintenance. Also other departments share data with the EMS and vice-versa, as illustrated in figure 5.1.

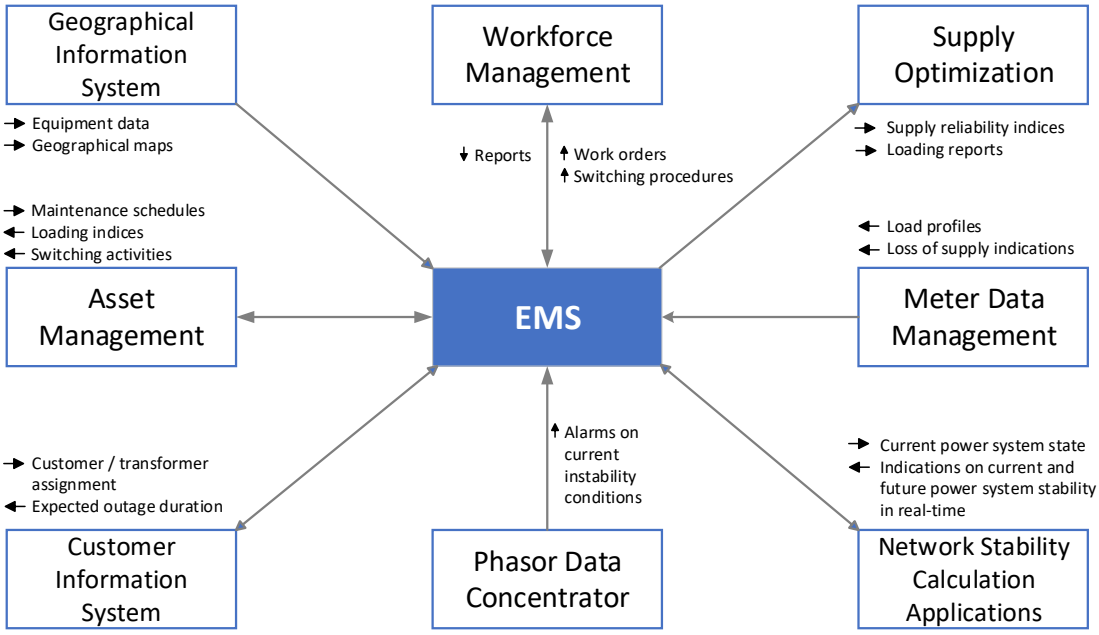


Figure 5.1: Scheme of the relations between the EMS and other grid operation departments

5.2 SCADA

Supervisory Control and Data Acquisition (SCADA) is the combination of telemetry and data acquisition via cable or radio communication system to monitor/operate a set of devices from a centralised location securely and efficiently. In the past, the control centre was directly connected to power substations, where personnel was present to operate the active assets, such as circuit breakers. With SCADA, the active equipment in the substations is controlled from a centralized location, without the necessity of local staff. SCADA systems are based on microprocessors and communication technologies and operate in the time scale of seconds to hours.

The main functions of SCADA systems are:

- Data acquisition, condition and monitoring;
- Event/disturbance detection and alarming;
- Remote control;
- Data logging;
- Report generation;

The SCADA control room is the place where all the information registered in the substations is displayed to the operators, that can remotely operate the active equipment. It is usually highly secured and inaccessible, and operators are highly qualified and have to observe strict rules.

In a generic SCADA system for industrial application, the structure is as follows. The assets (that in the case of a power grid are the equipment in the substations, such as transformers, circuit breakers, busses, etc...) are connected to **Remote Terminal Units (RTUs)**, that through sensors (analog or digital) translate the information into the protocol used at the substation level. The information is then sent to the **Master Terminal Unit (MTU)**, which provides a human-machine interface to the system in the control room. A communication network is used to connect the SCADA MTU to the RTUs in the substations.

Protocols for SCADA systems

DNP3 is one of the protocols used in SCADA systems. It is a leader-follower protocol, mostly used in North America, Australia and South Africa. Can send different classes of packets, with different importance levels. It is based on the Modbus protocol, but with enhanced features.

Then, IEC 60870-5-101 (usually referred to as **101**) is the standard used for basic remote control tasks. It has mostly been replaced by IEC 60870-5-**103** and IEC 60870-5-**104**, which is an extension of the IEC101 for data transport over TCP/IP stack. Also this protocol is leader-follower and used to be the most used in Europe and the Middle East. IEC103 is specifically used for protection devices, that act as followers, while the station controller acts as leader. IEC 60870-6 is an inter-control Center Communications Protocol, used to exchange information between different control centers, over wide area networks.

The most important modern SCADA protocol is the **IEC 61850**, which will be thoroughly presented in the following chapter. Other complementary protocols are the IEEE C37.118, for Synchronized Measurements Technology, and IEEE 1588 (PTP) for Precision Time synchronization. In reality, different protocols can be used at the same time for different purposes in a SCADA system.

5.3 Digital substations and IEC 61850 protocol

In the last years, a process of digitalization of the power substations has taken place. In a digital substation, the electrical assets are equipped with ICT devices that digitise the signals at the source, enabling digital communication, regulated by the IEC 61860 protocol. Moreover, new typologies of non-conventional sensors are used. Figure 5.2 illustrates the architecture levels of a digital substation. As pictured in figure5.2, the substation is also connected to the control centre, to enable the control of the power system at a broader level.

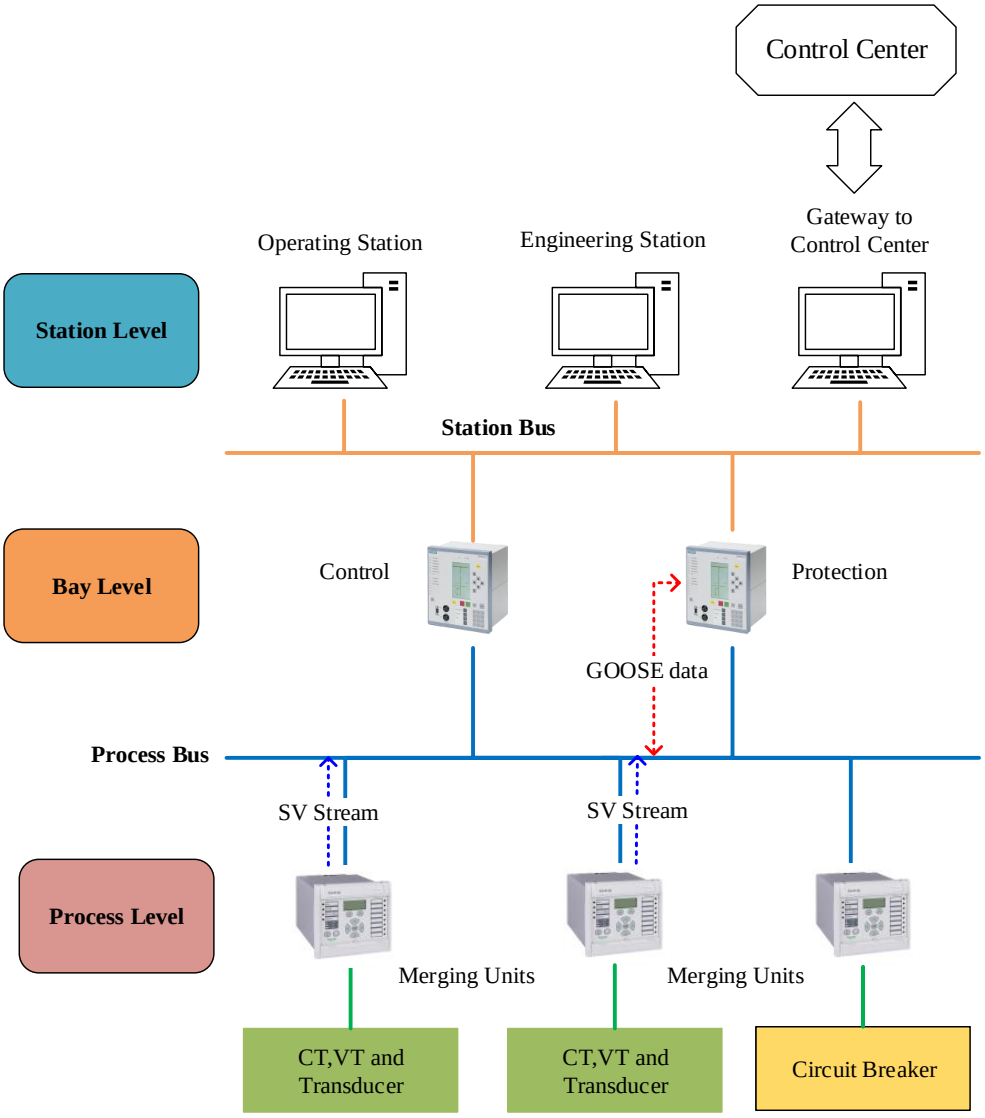


Figure 5.2: Scheme of the architecture levels of a digital substation

The main advantages of digital substations are:

- Interoperability: the IEC 61850 is a standard that allows purchasing components from different manufacturers, enabling competition and decreasing the costs;
- Electrical signals are digitized right at the source, using non-conventional sensors;
- improved measuring accuracy and performance;
- ease of device configuration, real-time performance conditioning;
- Maximized reliability and availability, extensive troubleshooting;
- Increased safety and lower environmental impact;
- Improved cost of ownership by eliminating copper wires (by 80%), space and massive hardware components.

IEC 61850 protocol

The IEC 61850 is a standard for SCADA communication that is bringing open, interoperable systems and flexible architectures to the substation automation domain. It is standardized in the function objects, meaning that there is a standardized definition and description for every component. The protocols are based on Ethernet, but specific switches with priority tagging are required. IEC 61850 allows for a reduction in the amount of cabling used, since the signals are digitized at the source.

Figure 5.3 illustrates the IEC 61850 communication stack. It is possible to observe that within the IEC 61850, different sub-protocols are used for different aims. The three main sub-protocols are: SV, GOOSE and MMS. SV and GOOSE are simpler, as they only present layers 1 and 2 and then directly the application layer.

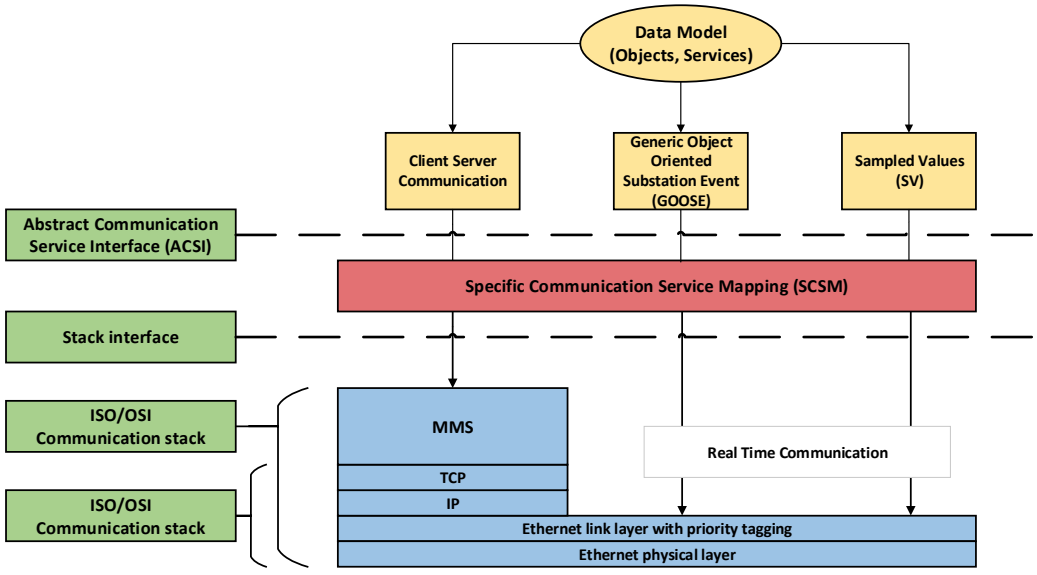


Figure 5.3: IEC 61850 protocol stack layers

Sample Values (VS) are used to transmit digitized instantaneous values of power system quantities. It is based on publisher/subscriber principle, meaning that one device (sender) publishes information and all the subscribed devices receive it, without the necessity of routing. The simplicity of the protocol makes it efficient.

Generic Object Oriented Substation Event (GOOSE) is used for status interactions, such as commands and alarms, between IEDs in a substation. Since GOOSE messages are used for alarms and security equipment, they need to have low latency. Within the GOOSE standard, there are different specific applications, detailed in the IEC 61850-7-2.

As previously seen, SV and GOOSE lack layer three and above, therefore they cannot be routed outside the substation. However, it is possible to work around this limitation through specific devices, described in the IEC 61850-90-2 protocol, enabling communication between different substations. For example, routable GOOSE messages can be used to communicate by circuit breakers located on different sides of a power line. When both circuit breakers individuate a fault, they are activated and the line is disconnected.

Manufacturing Messaging Specification (MMS) is a more complex protocol. It has an IP stack, hence it can be routed. It is used for non-time-critical functions, such as download and upload of configuration, parameter setting and monitoring of parameters. It is based on client-server principle.

IEC 61850 uses self-descriptive tags, meaning that the variables are more readable. For example, the variable "Health of circuit breaker 1" is tagged as "Relay1/XCBR1$Health% - stVal". As a comparison, in the Modbus and DNP3 protocol, the same variable is respectively labelled as "Register 41003" and "Obj 30 Var 1 Index 3". However, a possible drawback of this feature could be that it makes IEC 61850 more vulnerable to reconnaissance in cyber attacks.

5.4 Synchronized Measurement Technology (SMT)

SCADA systems present some drawbacks. In fact, the data acquisition latency is between 1 to 4 seconds. Hence, the SCADA system provides snapshots of the state of the grid, but it is not able to capture faster dynamics. Moreover, the communication of the data is slow and unreliable. The data have to be computed (in up to a few minutes) and then visualized by the operators. When an alarm occurs, usually the operators can rely on a set of pre-calculated action plans for a set of situations. However, taking into account all the mentioned steps, a considerable time occurs between a fault and a manual action to clear it.

With the increasing complexity of the grid, with high renewable penetration, lower inertia and therefore faster dynamics, SCADA systems could not be sufficient anymore to ensure its security. In this context, a new technology for power system monitoring is emerging: Synchronized Measurement Technology (SMT).

Synchronized Measurement Technology (SMT) is a collective technology used to monitor system-wide dynamics in real-time. It is based on Phasor Measurements Units (PMUs), which are measurement devices able to sample fifty times per second (or also more). As a comparison, SCADA provides only one measurement every few seconds (up to 4s). The information provided by PMUs is then merged by Phasor Data Concentrators (PDC) and synchronized through a precise time synchronization source. Finally, Information and Communication Technology infrastructure is used to transport the data streams to the control centre. Figure 5.4 illustrates the scheme of a SMT system. SMT can deliver precisely time-synchronized system-wide measurement in real-time, and therefore can be used for Wide

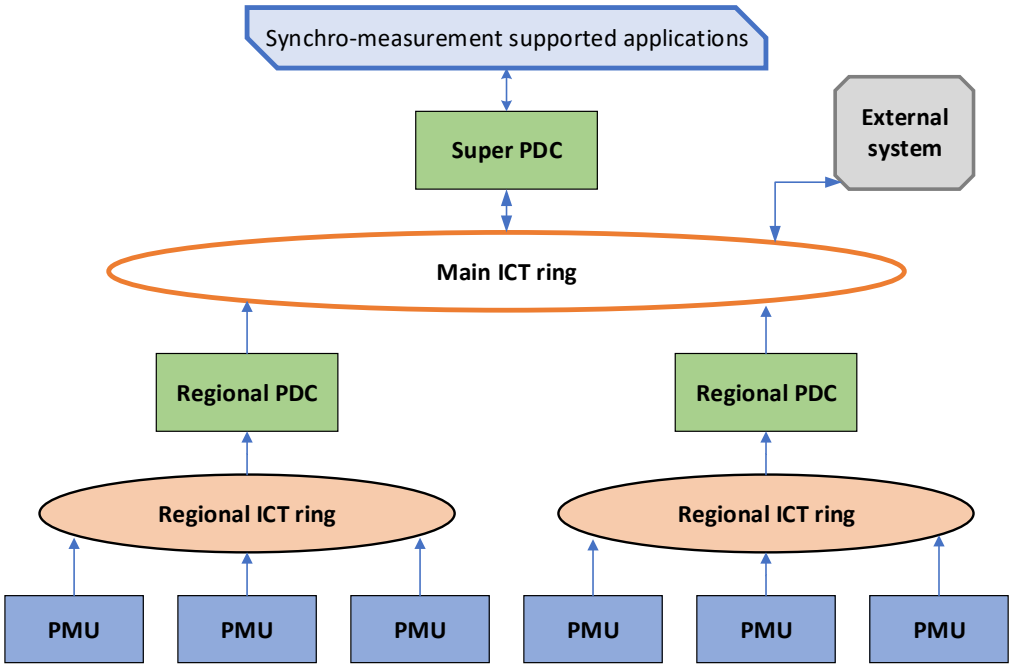


Figure 5.4: Scheme of the architecture of a SMT system

Area Monitoring. Protection, and Control systems (WAMPACs).

5.4.1 Phasor Measurement Unit (PMU)

PMUs provide time-synchronized measurements of:

- 3 Phase and Symmetrical Components Voltage Magnitude and Angle (Synchrophasor);
- 3 Phase and Symmetrical Components Current Magnitude and Angle (Synchrophasor);
- Frequency (deviation from nominal in mHz);
- Rate-of-change-of-frequency (ROCOF in Hz/s);
- Analog user defined signals (e.g. rotor speed, sampled control signal or value);
- Digital user defined data (e.g. breaker discharges, switch gear position).

These data are provided with a reporting rate of 10, 25, or 50 times per second. The protocols for measurement, estimation requirements and data transfer are specified in the IEEE C37.118 standard.

Voltage and current are provided as synchrophasors, which are rotating vectors with magnitude and phase with respect to the time reference. The vector, known as phasor, is the mathematical transformation of a sinusoid (that could be an AC current or voltage signal) in a complex number, as in equation 5.1:

$$X = X_{Re} + jX_{Im} = \frac{X_m}{\sqrt{2}} e^{j\Phi} = \frac{X_m}{\sqrt{2}} (cos\Phi + jsin\Phi) \tag{5.1}$$

This transformation is true only when the signal is a pure sinusoid. In presence of noise, as usually happens in real voltage and current signals, only the main wave is considered.

The measured synchrophasor rotates according to the frequency. Since the reference frame rotates at nominal frequency, if the measured synchrophasor appears still, it means that the measured signal is at nominal frequency. On the other hand, if the synchrophasor moves, it means that the frequency of the signal is different from the nominal one. Figure 5.5 gives a visual representation of the explained mechanism. The vertical lines, (correspondent to $t_0, t_1, ...$) represent the reference frame, at nominal frequency, while the green sinusoid is the AC signal. The intersection of these gives the direction of the synchrophasor, that in the illustrated case is rotating clockwise, meaning that the frequency of the signal is higher than the nominal one.

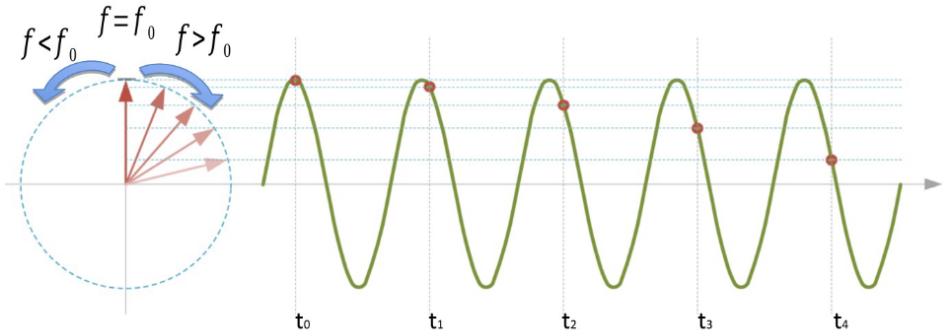


Figure 5.5: Graphical representation of a synchrophasor

In reality, the measurement of frequency and phasors is complex, to the point that it is often referred to as "estimation", instead of "measurement". In summary, the voltage and current signals are first filtered and synchronized, through the reference provided by a GPS receiver. Then, the frequency is estimated, and consequently, also the phasor is obtained. This is converted into a symmetrical component and sent. Figure 5.6 gives a more detailed representation of the described process.

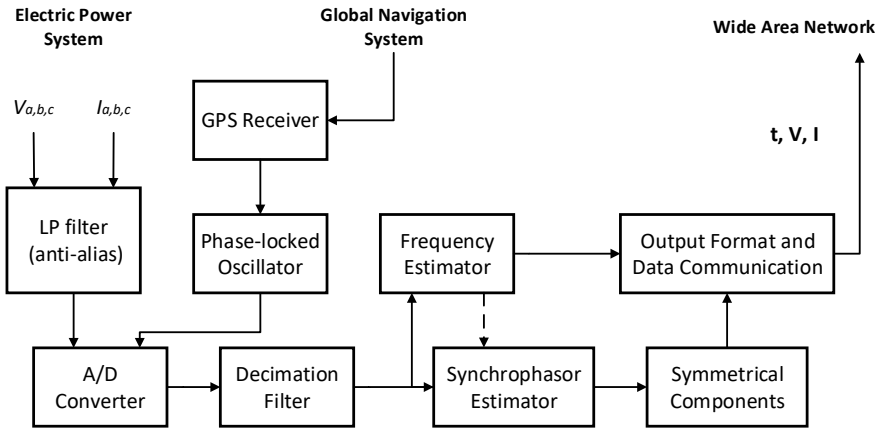


Figure 5.6: Scheme of the process of synchrophasor estimation

During the process of phasor estimation, different sources of errors can reduce the accu-

racy of the result. In particular, the following features should be taken into account:

- quality of synchronism: for example, an error of 1 μs corresponds to a phase angle error of 0.02°;
- quality of PMUs, that is the quality of the measurement itself;
- quality of instrumentation channels, that is the quality of the sensors.

These sources of error can be incorporated into the sinusoid representation, as in equation 5.2:

$$x(t) = X[1 + \varepsilon_a(t)] \cdot cos[2\pi f_0 \cdot (1 + \zeta)t + \varepsilon_p(t) + \Phi] + \varepsilon_h(t) + \varepsilon_n(t) \tag{5.2}$$

where: X is the average amplitude; $\varepsilon_a(t)$ are the amplitude fluctuations; f_0 is the nominal frequency; $\varepsilon_p(t)$ represents the phase fluctuations; Φ is the initial phase; $\varepsilon_h(t)$ represents the narrow-band disturbances, such as harmonics and out-of-band interharmonics; $\varepsilon_n(t)$ represents the wideband disturbances, as noise and DC decaying offset.

The accuracy of the synchrophasor estimation is expressed by the Total Vector Error (TVE) 5.3, which indicates the maximum deviation from the real measurement:

$$TVE(n) = \frac{|x_a(n) - x_m(n)|}{|x_a(n)|} \tag{5.3}$$

where x_a is the true phasor and x_m is the measured one. Figure 5.7 gives a graphic representation of the TVE, where the blue circle represents the area where the error is smaller than the allowed one. The TVE should be within 1%, while the dynamic error can be within 3%.

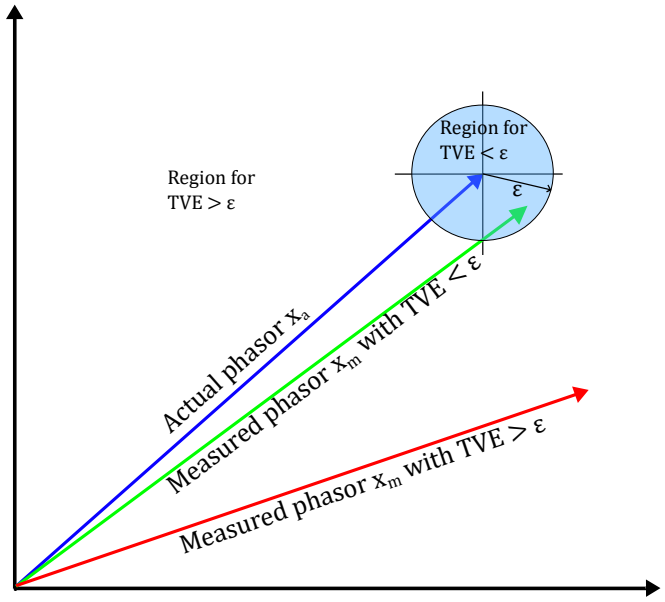


Figure 5.7: Graphic representation of the maximum allowed TVE

5.4.2 Time synchronization

Time synchronization is a fundamental part of SMT, as the synchronism error is directly reflected in the phasor estimation accuracy. There are different possible time-synchronization sources:

- Satellite time synchronization: GPS, GLONASS, Galileo and BeiDou. These are accurate, but all prone to spoofing.

- Wired time synchronization: Network Time Protocol (NTP), which is old and inaccurate, and the IEEE 1588 - Precise Time Procol (PTP), which is accurate below 60ns range.

In the IEEE 1588 - Precise Time Procol (PTP), the time signal is received from a source, such as GPS, and sent to the devices over Ethernet, taking into account the communication delay.

5.4.3 Phasor Data Concentrator (PDC) and Wide Area Communication Infrastructure

The Phasor Data Concentrator (PDC) receives the data from PMUs and other PDCs, time-aligns and sends them to higher applications. It also serves as data storage.

The Wide Area Communication infrastructure enables the communication of data between substations and the control centre. It is an IP-based Wide Area Network (WAN) that uses fiber optic, is arranged in a ring topology, and is based on advanced MPLS/IP or Carrier Ethernet protocols. Usually, TSOs develop their own WAN, to specialize it for the specific application.

5.5 Takeaway

WAMPAC fills the gap between local protection, which has a fast response time (within 100 ms) but is only local, and SCADA, which can operate at a global level but has a long response time. Moreover, thanks to their short sample time, PMUs allow observing the dynamic behaviour of the system, and not only static snapshots. Figure 5.8 summarizes the protection features of WAMPAC and the correspondent time frame.

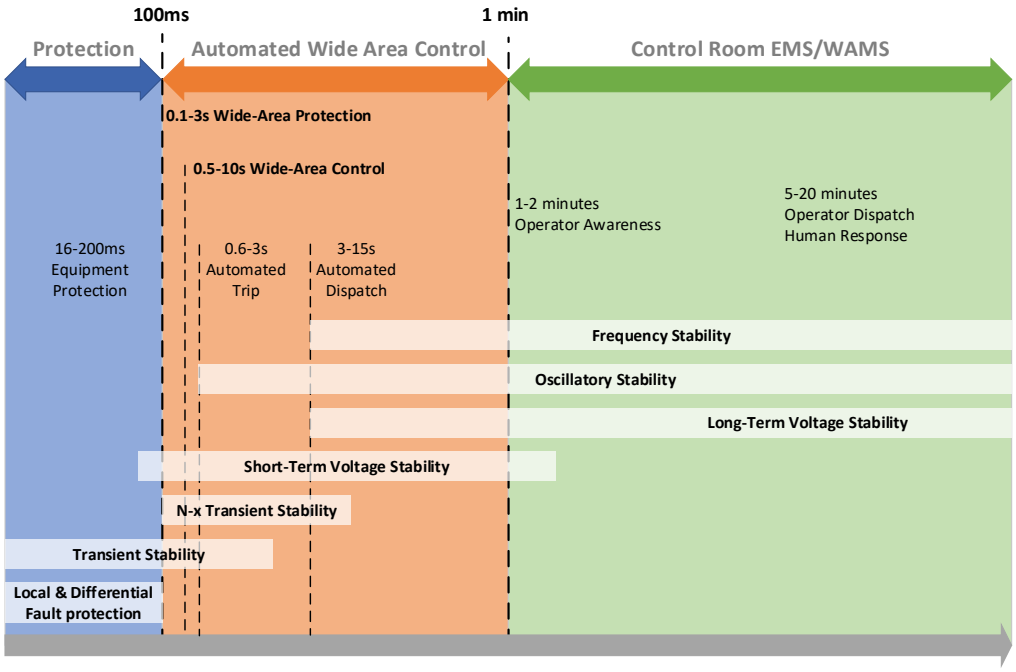


Figure 5.8: Summary of WAMPAC protection features, organised according to their time-scale

As a result, WAMPAC can be used to improve the dynamic performance of the grid, reduce the failure probability and reduces the impact of disturbances, thanks to a faster restoration. These benefits are therefore shared between all the stakeholders: system operators, generation, transmission and distribution. In the future, SMT could be used to create a Large Scale Wide Area Monitoring System (LS-WAMS), that involves multiple TSOs. In this sense, different designs are being studied, concerning a centralized system, with only one LS-WAMS, or a decentralized system with multiple coordinated LS-WAMS.

6. Data-driven distribution systems planning

This chapter introduces generative AI techniques for creating synthetic time-series data to support distribution system planning.

The learning objectives are:

(i) Evaluate key generative models: GMMs, VAEs, and GANs.

(ii) Describe metrics for data quality.

(iii) Compare model performance.

6.1 Introduction

In the context of modern energy systems, data-driven approaches are becoming increasingly central to the planning and operation of distribution networks. With the proliferation of sensors, smart meters, and advanced monitoring technologies, vast amounts of data are now available. However, using this data effectively needs advanced modeling methods that can evaluate the complex and changing nature of power distribution systems. This lecture introduces the foundational concepts involved in such planning, including generative modeling techniques for synthetic data creation and evaluation, along with flow and diffusion models. These tools play a pivotal role in supporting robust decision-making in distribution systems planning.

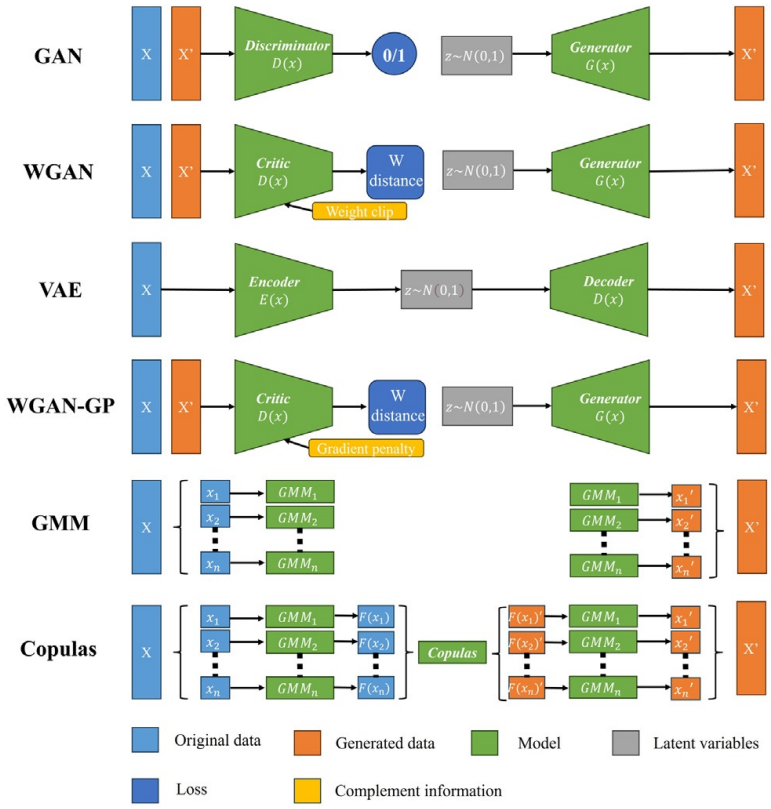


Figure 6.1: Illustration of the construction of six models and modeling approach highlighting the input (or original) data and the generated data, as well as each model's component, including the loss function and the latent variables (or space) if the model considers one. **??**

6.2 Generative AI for Synthetic Data Generation

Generative AI refers to a class of machine learning models designed to produce new, realistic data samples, such as text, images or time-series data. These models, including variational autoencoders (VAEs), generative adversarial networks (GANs) and others, have become valuable assets across various industries due to their ability to learn complex data distributions and generate novel outputs. A key strength of generative models is their ability to condition outputs based on desired parameters, making them adaptable for specific applications. In the energy sector, particularly for distribution systems planning and operation, this capability opens up opportunities to generate synthetic load profiles, fault scenarios and future demand patterns that are critical for robust system design.

6.2.1 Time-Series Energy Data Generation

There is a wide range of generative models available for time-series energy data generation, each offering different levels of complexity and architectural features. As summarized in Figure 6.1, these models vary in their approach to data representation, latent space formulation and loss functions. For example, gaussian mixture models (GMMs) model each time step as a separate random variable and construct independent distributions for each, with sampling done across all steps to create a complete real load profile (RLP). Gaussian mixture copula (GMC) models build upon GMMs by first transforming marginal distributions into

cumulative probabilities and then modeling joint behavior using a multivariate gaussian distribution. In contrast, deep generative models like GANs, WGANs, and VAEs introduce more sophisticated architectures. GANs employ adversarial training where a generator learns to create data indistinguishable from real samples by a discriminator. VAEs use a probabilistic framework to encode and decode data through a latent space, optimizing reconstruction and divergence losses.

Each model's performance and suitability depend heavily on the characteristics of the data being modeled. For instance, simpler models like GMMs may suffice for stationary or low-dimensional data, while high-variance or non-linear temporal data may benefit more from deep models like GANs or VAEs. Therefore, selecting the appropriate model must be guided by the type and features of the available data. Evaluating these distinctions is essential for generating generative AI effectively in distribution systems planning, ensuring that generated synthetic data accurately reflects the real-world complexities of energy consumption and system behavior.

6.2.2 Gaussian Mixture Models (GMMs)

GMMs are a robust and widely used statistical tool in machine learning. They are particularly effective in modeling complex, multi-modal distributions by representing them as a weighted sum of simpler gaussian components. This capability makes GMMs suitable for approximating distributions where the underlying data exhibit multiple modes or clusters. Mathematically, a GMM is defined as:

$$p(\theta) = \sum_{i=1}^{K} \pi_i \mathcal{N}(\mu_i, \Sigma_i) \tag{6.1}$$

where $\theta = \{\mu_i, \Sigma_i\}$ represents the set of parameters for each component, π_i are the mixing coefficients such that $\sum \pi_i = 1$, $\mathcal{N}(\mu_i, \Sigma_i)$ denotes the gaussian distribution with mean μ_i and covariance Σ_i and K is the total number of components in the mixture.

In the context of time-series energy data generation, we fit a separate GMM model for each time step. Each time step is considered an independent random variable, and a GMM is used to estimate its probability distribution. One of the main limitations of this approach in time-series modeling is that it disregards the temporal correlation between successive time steps. Since a separate model is built for each time step, the joint temporal structure across the series is not captured. This can limit the realism and accuracy of the generated synthetic sequences.

6.2.3 Variational Autoencoders (VAEs)

VAEs are a class of generative models designed to learn the underlying probability distribution of a dataset and generate new, realistic samples from it. VAEs are particularly effective for structured data such as time-series, where they can learn compact latent representations that capture the essential features of the input. A VAE is composed of two main components:

- Encoder: Maps the input data l to a latent space represented by variables z. This is modeled as a probabilistic distribution $Q_\phi(z|l)$, typically assumed to be Gaussian.
- Decoder: Reconstructs the input data l from the latent representation z by modeling $P_\theta(l|z)$.

The training objective of a VAE is to maximize the evidence lower bound, which can be expressed as the following loss function:

$$\mathcal{L}_{\text{VAE}}(\theta, \phi; l) = \mathbb{E}_{z\sim Q_\phi(z|l)}[\log P_\theta(l|z)] - \text{KL}(Q_\phi(z|l) \| P(z)), \tag{6.2}$$

where the loss function consists of two terms:

- Reconstruction loss: $\mathbb{E}_{z\sim Q_\phi(z|l)}[\log P_\theta(l|z)]$ encourages the decoder to accurately reconstruct the original data from the latent representation.
- Regularization loss: $\text{KL}(Q_\phi(z|l) \| P(z))$ is the kullback-leibler (KL) divergence, which regularizes the learned latent distribution to be close to the prior $P(z)$, typically a standard normal distribution.

By learning both the encoding and decoding processes, VAEs provide a principled way to generate new samples and explore the latent space of the data. They are especially useful in applications requiring structured synthetic data, such as time-series energy profiles in distribution systems planning.

6.2.4 Comparison of Generative AI Models for Time-Series Generation

When applying generative AI models to time-series data, particularly for energy consumption profiles, it is essential to understand their strengths, limitations and the quality of the generated outputs. While simple models such as GMMs are computationally efficient and powerful in approximating marginal distributions, they have significant limitations. One key drawback is the inability to accurately capture high-energy consumption profiles due to their independent modeling of each time step. As a result, part of the dynamic information especially temporal correlations and peak events may be lost. Visual inspection is helpful but subjective. To objectively evaluate model performance, we must assess the generated RLPs based on critical features relevant to distribution systems planning, such as peak magnitude, occurrence time, and overall consumption variance. The original data and synthetic RLPs generated by six different models for the NL-traf and UK-traf test cases are illustrated in Figure 6.2. One of the primary applications of RLPs is supporting the planning and operation of distribution systems, where it is vital to correctly estimate peak power consumption and its timing. Such accuracy improves the reliability of network planning by accounting for worst-case scenarios. In the NL-traf test case, the real data shows peaks around 9 AM and 8 PM. In the UK-traf test case, peak usage is generally observed around 8 PM.

All generative models replicated these temporal patterns successfully, demonstrating their capability to learn and reproduce critical load behaviors. However, GMMs exhibit a significant limitation: they tend to generate load profiles with an average daily energy consumption. This is evident in Figure 6.2, where the GMM-generated RLPs exhibit uniform coloring, indicating minimal variation. This is attributed to the fact that GMMs model each time step independently, without considering temporal dependencies, thus failing to reproduce realistic peaks and variability. The performance of generative models varies depending on the time resolution and the complexity of the input data. While GMMs offer simplicity and robustness, they are limited in capturing time-series dynamics. On the other hand, deep generative models like GANs and VAEs are capable of modeling temporal dependencies, making them more suitable for dynamic and high-resolution datasets. GMMs perform well in capturing marginal statistics (e.g., low wasserstein distance (WD)) but fail to preserve temporal structure, resulting in poor MSE and low variability. GAN-based models, such as

WGAN and WGANGP, better capture time correlations and peak shapes, yielding more realistic profiles. VAEs strike a balance between reconstruction quality and regularization, effectively maintaining temporal coherence and variability. Common evaluation metrics such as wasserstein distance (WD), jensen-shannon divergence (JS), kolmogorov-smirnov distance (KS) and maximum mean discrepancy (MMD) help quantify statistical similarity between distributions. However, these metrics are not absolute, particularly in the context of time-series data. They often fail to capture essential aspects like time-step correlation, leading to inconclusive or misleading assessments.

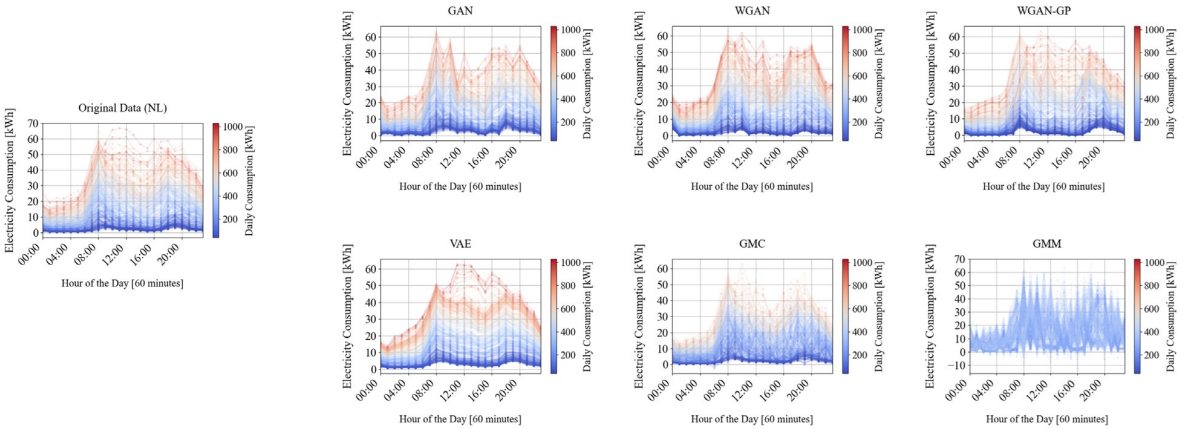


Figure 6.2: Right side: Results generated by six models for two transformer-level test cases. Left side: Original data. The color of the RLPs corresponds to the sum of daily consumption. **??**

While GAN-related models tend to outperform others in several performance categories, the results are not definitive across all metrics and time resolutions. Thus, model selection should be context-driven, factoring in application-specific priorities such as peak load prediction, daily consumption trends, and operational reliability. A comprehensive assessment should combine quantitative metrics, temporal structure evaluation, and visual inspection.

6.2.5 Synthetic Data Accuracy Quantification

Ensuring synthetic data is sufficiently accurate for its intended application is essential. Quality can be assessed from three key perspectives:

- Fidelity, which measures how closely the synthetic data replicates the statistical properties of the real data using metrics like WD, KL or JS divergence.
- Utility, which evaluates how well the synthetic data supports specific downstream tasks.
- Privacy, which considers the risk of re-identifying individuals from synthetic samples, often quantified using differential privacy-based metrics.

6.2.6 Flow-Based Models

Flow-based models are a class of generative models that transform a simple probability distribution (e.g., gaussian) into a complex one by applying a sequence of nested functions. The

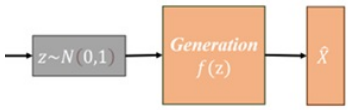


Figure 6.3: Operation function

transformation is defined as:

$$f = f_1 \circ f_2 \circ f_3 \circ \cdots \circ f_K$$

or equivalently:

$$f = f_1(f_2(f_3(f \ldots (f_K))))$$

Imagine operation function which is depicted in Figure 6.3: Here, z is described by the distribution p_z, f is a bijective function and x is described by the distribution p_x and the relationship between distributions p_z and p_x is

$$p_x(x) = p_x(x = f(z)) = p_x(x = f(z \sim p_z))$$

and using the change of variable theorem, $p_x(x)$ is expressed as:

$$p(x) = \pi(z)\left|\det \frac{dz}{dx}\right| = \pi(f^{-1}(x))\left|\det \frac{df^{-1}}{dx}\right| \tag{6.3}$$

and this becomes

$$p_x(x) = p_z\left|\det\left(\frac{\partial f^{-1}}{\partial x}\right)\right|$$

This relationship is critical because it allows us to compute the exact likelihood of data x given a transformation f. Suppose the transformation f is parameterized by θ, i.e., f_θ. We seek to optimize θ by maximizing the log-likelihood:

$$\theta = \arg\max \log(p_x(x)) = \log(p_z) + \log\left(\left|\det\left(\frac{\partial f^{-1}}{\partial x}\right)\right|\right), \tag{6.4}$$

where $\log(p_z)$ is known and the jacobian determinant term $\log\left(\left|\det\left(\frac{\partial f^{-1}}{\partial x}\right)\right|\right)$ can be computed and with these results, flow model is build.

To model more complex distributions, we can compose multiple nested functions f_i, forming a nested transformation $f = f_1 \circ f_2 \circ \cdots \circ f_K$. In this extended formulation, the log-likelihood of a data point x becomes:

$$\log(p_x(x)) = \log(p_z) + \sum_{i=1}^{K} \log\left(\left|\det\left(\frac{\partial f_i^{-1}}{\partial z_j}\right)\right|\right) \tag{6.5}$$

This additive form is central to flow-based models, enabling them to learn expressive distributions while maintaining exact likelihood evaluation. To implement each f_i, we often use neural networks. A common solution is the use of coupling layers, which split and transform the input in a controlled, reversible manner.

6.3 Diffusion-Based Models

Denoising diffusion probabilistic models (DDPMs), or simply diffusion models, represent a new class of generative models that have gained significant popularity due to their ease of training and outstanding performance in generative tasks, particularly image synthesis. Diffusion models work by learning to reverse a gradual noising process applied to the data. Despite their remarkable success in image generation, standard DDPMs face challenges when applied to time-series data generation, particularly in high-resolution domains such as energy systems. The primary limitations of standard diffusion models for time-series generation include:

- Scalability to high-resolution data: Standard DDPMs struggle to efficiently process fine-grained data such as 1-minute resolution time-series due to the long sequence lengths and temporal dependencies.

- Marginal distribution mismatch: Inaccurate approximation of marginals can lead to poor modeling of key characteristics such as peak values, which are critical in energy-related applications.

To address these issues, we propose energydiff, a customized diffusion model architecture tailored to the specific needs of high-resolution, stochastic energy data modeling.

6.3.1 The Diffusion Process

The fundamental idea behind diffusion models is to define two processes: a forward diffusion process that gradually corrupts the data by adding gaussian noise, and a reverse process that learns to denoise the corrupted data to recover samples from the original data distribution. We begin with the original data x_0 that follows distribution $p_\theta(x_0)$ as illustrated in Figure 6.4. The forward diffusion process is defined as:

$$q(\mathbf{x}_s|\mathbf{x}_{s-1}) := \mathcal{N}(\sqrt{1-\beta_s}\mathbf{x}_{s-1}, \beta_s\mathbf{I}), \tag{6.6}$$

where $\beta_s \in (0,1)$ denotes the noise level at step $s \in \{1,\dots,S\}$. As s increases, the sample becomes progressively noisier. If $s = S$ is large enough, the data will be almost completely corrupted and is

$$q(\mathbf{x}_s|\mathbf{x}_0) \cong q(\mathbf{x}_s) = \mathcal{N}(\mathbf{0},\mathbf{I})$$

In the reverse process, we start with fully corrupted data $\mathbf{x}_S \sim \mathcal{N}(\mathbf{0},\mathbf{I})$, and aim to recover a sample from the original data distribution by learning a sequence of denoising transformations. This reverse process is parameterized by a neural network:

$$p_\theta(\mathbf{x}_{s-1}|\mathbf{x}_s) := \mathcal{N}(\mu_\theta(\mathbf{x}_s,s), \Sigma_\theta(\mathbf{x}_s,s)), \tag{6.7}$$

where μ_θ and Σ_θ are functions of s. Typically, only a single neural network is needed, which is also a function of s. The following algorithm outlines the DDPM sampling process from a trained model:

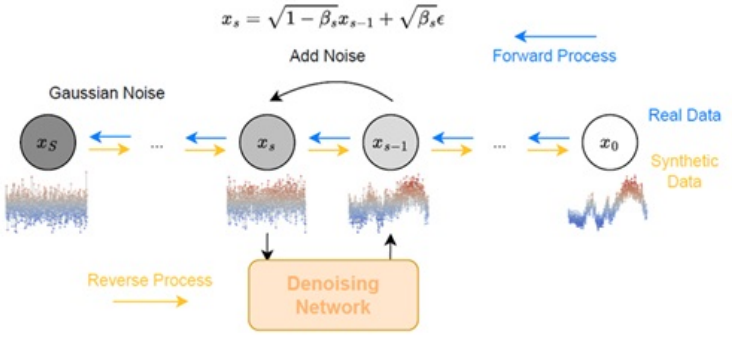


Figure 6.4: In the forward process, noise is injected into the true data at gradually increasing levels; in the reverse process, the noise is iteratively removed from noisy data.

Algorithm 1 Sampling new data from DDPM

Require: trained μ_θ, noise schedule $\{\beta_s, \alpha_s, \bar{\alpha}_s\}$
1: Initialize $s \leftarrow S$
2: Sample $x_S \sim \mathcal{N}(0, \mathbf{I})$
3: **while** $s > 0$ **do**
4: *Reverse:* Calculate $\hat{\mu}_{s-1} = \mu_\theta(x_s, s)$
5: Sample $x_{s-1} \sim \mathcal{N}(\hat{\mu}_{s-1}, \tilde{\beta}_s \mathbf{I})$
6: $s \leftarrow s - 1$
7: **end while**
8: **return** x_0

6.3.2 EnergyDiff

Modeling high-resolution time series data poses significant computational and memory challenges. For example, a daily electricity profile sampled at 1-minute intervals leads to a 1440-dimensional vector. Even relatively simple generative models, such as GMMs, can require over a million parameters to model such data effectively. This complexity escalates further with deep learning approaches, particularly when accurate marginal distributions (e.g., peaks) must be captured for tasks like energy demand forecasting or synthetic data generation. To address these issues, we introduce EnergyDiff, a diffusion-based model specifically designed for high-resolution energy time-series data. It builds on the standard DDPMs, retaining the same forward (noise injection) process while modifying the reverse (denoising) process with a structure tailored for time series modeling. The architecture of EnergyDiff is presented in Figure 6.5 and ey innovations of EnergyDiff include:

- Folding Block: To handle long sequences, the time series data is folded (grouped) into smaller segments before processing. A 1440-step sequence can be folded every $r = 3$ steps to reduce the input size by a factor of 3 as illustrated in Figure 6.6. This transformation helps reduce the sequence length while preserving local temporal correlations.

- Transformer-based Denoising Network: After folding, the sequence is passed through a series of Transformer blocks. These blocks are well-suited for modeling temporal dependencies and allow the network to learn complex sequential dynamics. Positional

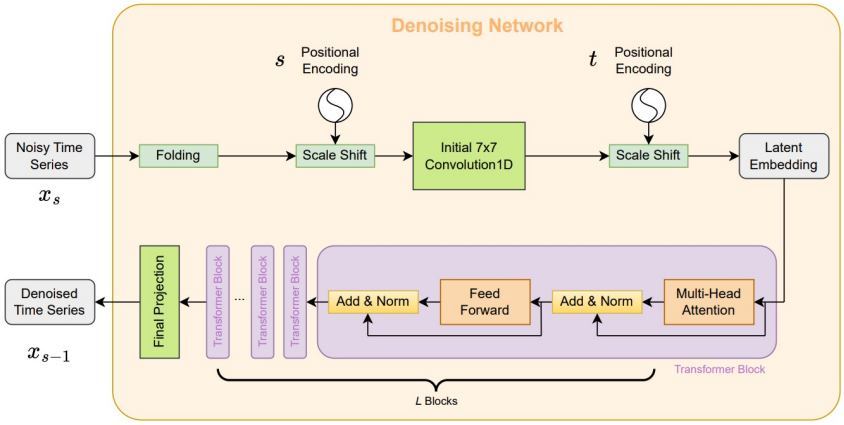


Figure 6.5: Denoising network architecture

encodings are injected to preserve time-step and diffusion-step information (denoted t and s).

- Projection Layer: To ensure effective information flow across the depth of the network, the outputs of the first and last transformer blocks are concatenated and projected to produce the final denoised estimate x_{s-1}.

- Positional Encodings: The architecture includes two positional encodings: one for the diffusion step s and one for the original time index t. This ensures the model understands both the temporal order and the stage in the diffusion process.

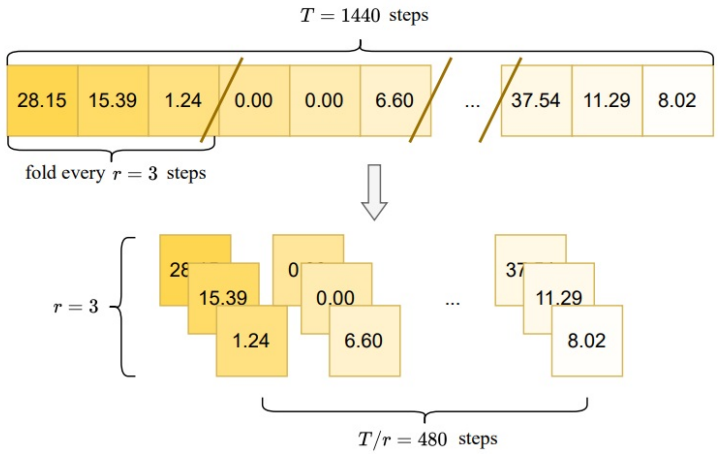


Figure 6.6: A long univariate time series of 1440 steps is folded into a shorter 3-variable (or 3-channel) multivariate time series of 480 steps, with a folding factor r = 3

The result is a model capable of generating highly realistic, high-resolution synthetic time series that respects temporal structure and peak behavior. EnergyDiff is particularly useful in domains like smart grid modeling, where time-step granularity and data realism are crucial.

6.4 Distribution System Operation and Planning

Distribution system planning is undergoing a paradigm shift as increasing levels of distributed energy resources (DERs), such as photovoltaic (PV) systems, create significant technical and operational challenges. To navigate these complexities, distribution system operators (DSOs) require robust frameworks that enable risk-informed decision-making before committing to costly network reinforcement projects. A data-driven planning framework is introduced for the early-stage technical assessment of medium-voltage (MV) distribution networks. The framework, depicted in Figure 6.7, is designed to identify network areas that may become technically constrained due to load and PV generation. These identified areas can then be subjected to more detailed techno-economic analysis, which is not the scope of this work.

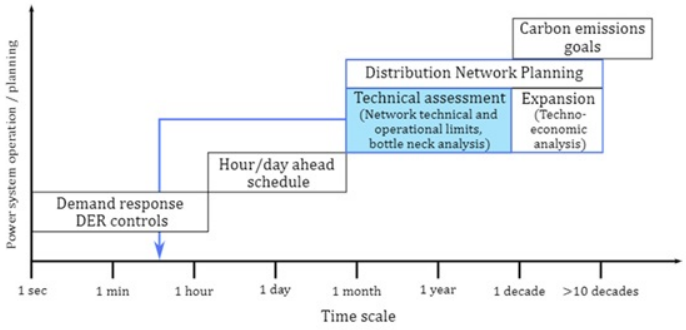


Figure 6.7: Framework for early-stage distribution network planning. The blue rectangle highlights the technical assessment step used to identify networks requiring further techno-economic analysis.

6.4.1 Modeling of Load and PV Generation

The proposed framework simulates load demand and PV generation at a 15-minute resolution to respect the concurrency between consumption and generation patterns. This temporal granularity is essential for accurately detecting technical violations such as voltage deviations or thermal overloads. Load profiles are forecasted considering future growth scenarios based on historical data and socio-economic factors. Simultaneously, PV generation is modeled using irradiance profiles characterized by the stochastic nature of weather patterns. These include the probability distributions of sunny, cloudy and overcast days in a specific geographical region. A comprehensive modeling and data processing pipeline is used to simulate thousands of probabilistic power flow (PPF) cases, capturing the stochastic nature of future irradiance and load scenarios. Fig. 6.8 illustrates the key steps in this process, from data preparation to scenario generation.

6.4.2 Critical Static Operating Regions and Nomograms

The concept of critical static operating regions is introduced to visually capture the technical feasibility of operating points defined by combinations of annual energy consumption and installed PV capacity. These are depicted in Figure 6.9, with regions defined as safe, caution,

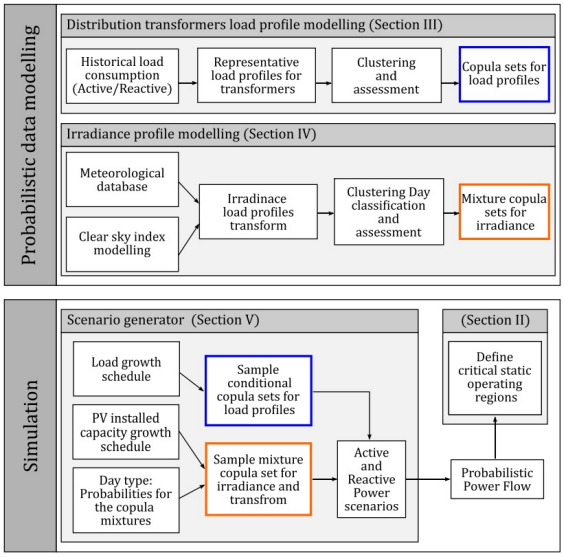


Figure 6.8: Proposed approach for the study of MV distribution network, under stochastic conditions.

etc. The caution regions can be further examined using irradiance-dependent ternary plots,

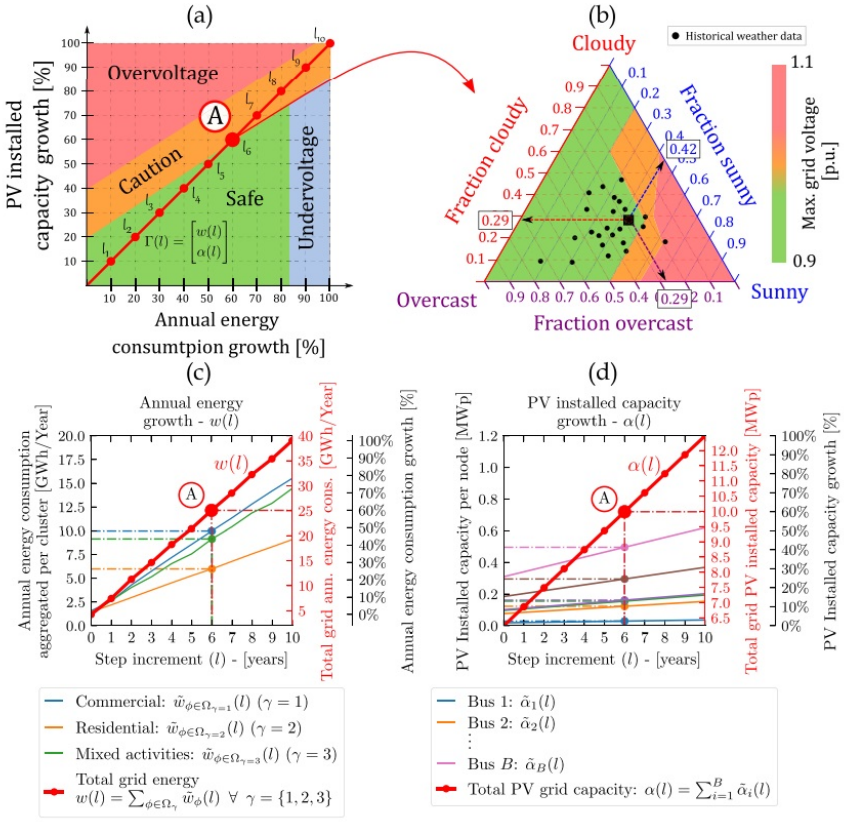


Figure 6.9: Visualization of static regions for the voltage technical limits in the network

as shown in Figure 6.9. Each point reflects the fraction of sunny, cloudy, and overcast days

in a month. For example, a summer month in the Netherlands might contain 42% sunny, 29% cloudy, and 29% overcast days. The colored contours in the plot reveal the overvoltage risk under such conditions. It further displays the annual growth trajectories for energy consumption $w(l)$ and PV installed capacity $\alpha(l)$, which together form the parametric path $\gamma(l)$. A specific year's condition, such as year six, is represented by a point along this trajectory (e.g., Point A).

Using the scenario simulations, the framework enables visualization of critical frontiers representing technical operating boundaries under different risk tolerances. Figure 6.10 show these frontiers for 0%, 10%, and 25% risk scenarios. The purple curve in each subplot represents the baseline (0% risk). As an illustrative example, under 40% annual energy

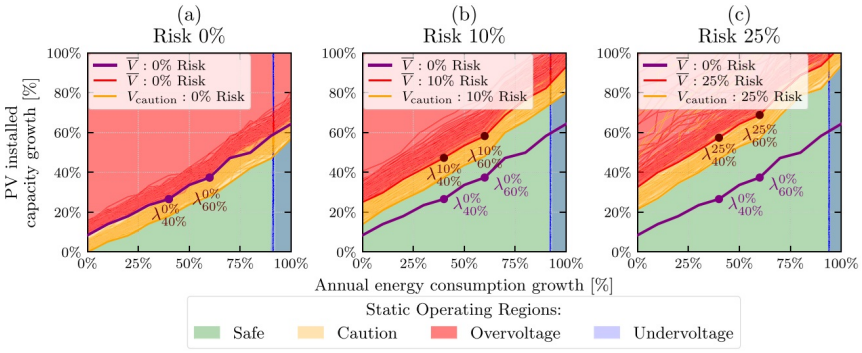


Figure 6.10: Overlay of collection of critical frontiers generated by different irradiance conditions for different levels of risk

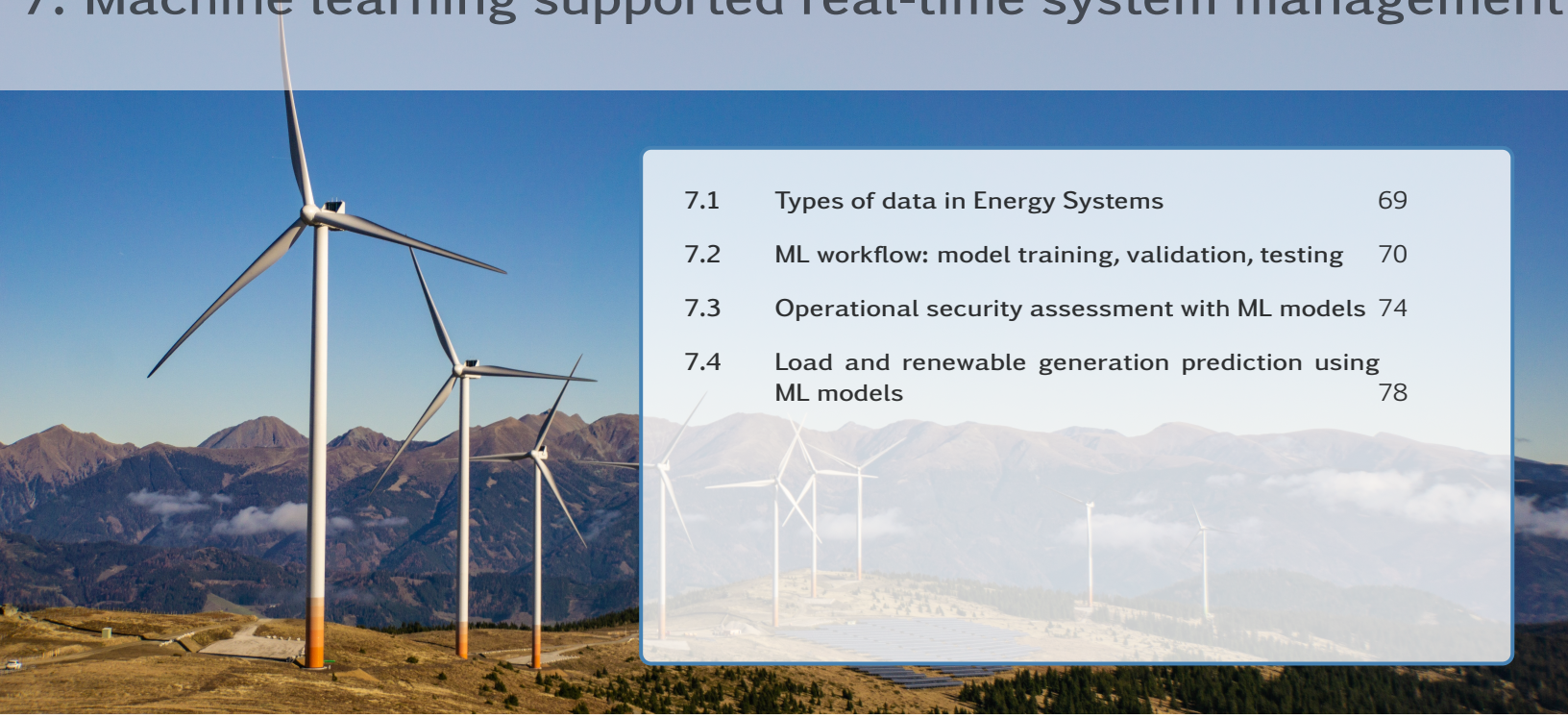
growth, allowing 5% operational risk increases the maximum PV hosting limit from 26.5% to 42.3%, a 15% gain in hosting capacity. The proposed framework combines high-resolution simulations, probabilistic modeling, and visual analytics (nomograms, ternary plots) to support early-stage planning of distribution networks under uncertainty. By quantifying risk and identifying Critical Static Operating Regions, DSOs can better assess hosting capacity and strategically defer or target grid reinforcement. This risk-aware approach provides a powerful tool for maximizing the utility of existing infrastructure while supporting renewable integration goals.

6.5 Conclusions

- Generative AI models can help overcoming the lack of (smart meter) energy data due to privacy constraints.
- We need to be careful with the metrics we use to evaluate the quality of synthetic generated data (mind the fidelity, utility and privacy).
- Flow models take advantage of nesting simpler function to map complex functions, similar as to add more layers in MLP. Generally, they showed good performance, even to generate outliers.
- Simple operations like folding and encoding the data reduce the need for deeper and more complex neural networks in case of larger time series data (the power of ChatGPT lays here).

- (Easy) Visualization (in general, explainability) of such large amount of data remains as a challenge. We aimed to address this by proposing the Critical Static Operating Regions.
- Planning considering accurate (generation, consumption) data modelling allows reducing uncertainty when allowing risk.

7. Machine learning supported real-time system management



This chapter deals with the main applications of machine learning models in energy systems. The learning objectives are:

(i) Describe machine learning methods for assessing power system security.

(ii) Describe machine learning methods for predicting generation and consumption.

7.1 Types of data in Energy Systems

This section addresses the notation and the types of data used in machine learning applied to energy systems. Additionally, noise, correlation, and imputation will be introduced.

Notation and types of data

Every data point is written as $x_{i,j}$, where j identifies the feature and i the sample/data point. X is usually a matrix and x a vector.

Different types of data are used in energy systems:

- Temporal data: the data points are expressed as a function of time, hence i represents time. Temporal data can have different resolutions, meaning that the time span between data samples can vary.

- Continuous data: can take on the value of any real number within a given range of real numbers.
- Categorical data: have a finite set of discrete values across the population of data [8].
- Discrete data: consists of distinct, separate values, often representing counts or integers. These values are typically finite and do not have intermediate values between them. For example, the number of units of energy generated or consumed.

Noise

"Noise" indicates the unwanted fluctuations of the signal, mainly caused by the measuring instruments. The intensity of noise can be assessed with the signal to noise ratio (SNR), that is the ratio between the power of a signal P_{signal} and the power of the background noise P_{noise}:

$$SNR = \frac{P_{signal}}{P_{noise}} \tag{7.1}$$

Noise usually has an expected value of zero.

Correlation

"Correlation" refers to a statistical relationship between two variables. The correlation between two random variables X_1 and X_2, with expected values μ_{X_1}, μ_{X_2} and standard deviations $\sigma_{X_1}, \sigma_{X_2}$, can be measured through the correlation coefficient $\rho_{X_1X_2}$:

$$\rho_{X_1X_2} = corr(X_1, X_2) = \frac{cov(X_1, X_2)}{\sigma_{X_1}\sigma_{X_2}} = \frac{E[(X_1 - \mu_{X_1})(X_2 - \mu_{X_2})]}{\sigma_{X_1}\sigma_{X_2}} \tag{7.2}$$

The correlation can be either positive or negative and can be observed through a graph, with the two variables on the axes. In case of correlation, the data points are distributed in a defined area, over a line. If no correlation occurs, the data points are distributed all over the graph.

Imputation

In case of missing features, it is possible to impute the data, instead of discarding the data point. Univariate imputation imputes values of the j^{th} feature dimension using only non-missing values in that feature dimension. The missing value can be imputed with a constant value, with the mean, median or most frequent value, or using the gradient of the previous points. Multivariate imputation algorithms use the entire set of feature dimensions.

7.2 ML workflow: model training, validation, testing

The principle of machine learning is to train a model based on sample data. The data used to train the model are known as "training data". The model can then be used to make predictions and decisions. There are several different techniques. One way to divide them is to distinguish **supervised learning** from **unsupervised learning**. In order to understand this classification, it is first necessary to define the following concepts: input and output.

A machine learning model can be seen as a box, with an input and an output. The **input** of a machine learning model is a matrix X of p **features**, which are individual independent variables, and n samples. The resulting matrix X is the following, where $x_{i,j}$ is the value of the j^{th} feature for the i^{th} observation, with $i = 1, \dots, n$ and $j = 1, \dots, p$. An example of input data in an Excel sheet is shown in figure 7.1

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}$$

	j1	j2	j3	j4
	wind speed	Temp	Time	Month
i1	3	20.5	01:00	1
i2	4	22.3	09:00	2
i3	4.5	25.1	08:00	4
i4	5	23.6	12:00	3
i5	6	28.8	05:00	5
i6	2	19.5	06:00	6
i7	8	18.1	11:00	7
i8	5	9	08:00	2
i9	4	5	03:00	3
i10	3.5	4	02:00	6

Figure 7.1: Example of input data

The **output**, also called **label**, is the result of the model. The output is denoted with y, a n-dim vector. The output can be continuous or discrete.

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

7.2.1 Models classification

As mentioned earlier, machine learning models can be distinguished into supervised and unsupervised. The difference is that in **supervised learning** there is a label (output) associated with the input, while in **unsupervised learning** the label is not present. The unsupervised learning technique that will be discussed in this lecture is **clustering**. The supervised learning techniques in this lecture are **regression**, which typically has continuous outputs, and **classification**, which has discrete outputs. The classification of the machine learning models is summarized in figure 7.2:

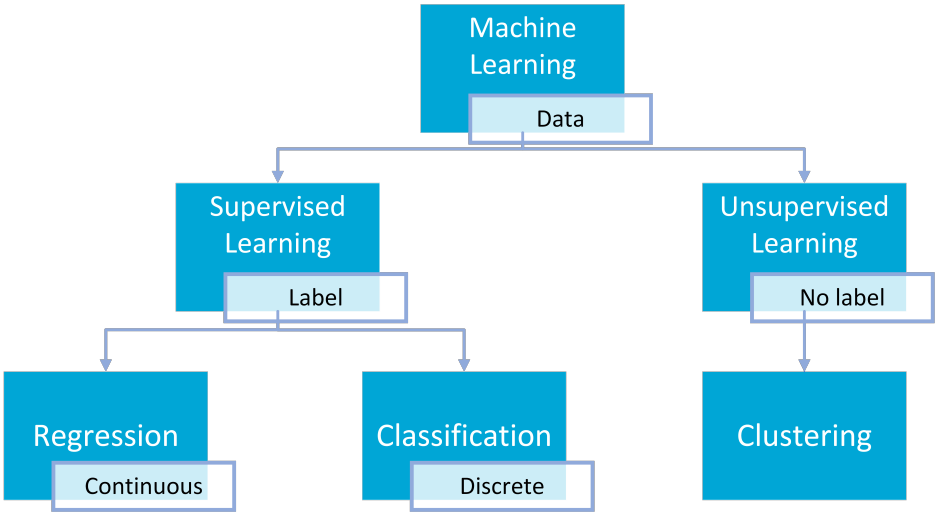


Figure 7.2: Classification of machine learning models

Unsupervised learning

In unsupervised learning the output is not present and it is referred to as "unsupervised" because there is no response variable that can "supervise" the analysis. Unsupervised learning aims to investigate the relationships between the variables or the observations [2].

One example of unsupervised learning is **clustering**, whose goal is to gather data that present some similarities in distinct groups. In this example, taken from the study of Yilmaz et al. [9], a customer segmentation for demand response is performed. Every customer is described by many features, such as energy consumption, employment, education and if he/she owns an electric vehicle, a PV system or a home battery. The model will find out if customers with similar characteristics can be grouped together. Figure 7.3 gives a visual representation in two dimensions of the described clustering problem. The names of the groups in figure 7.3 are not given by the model.

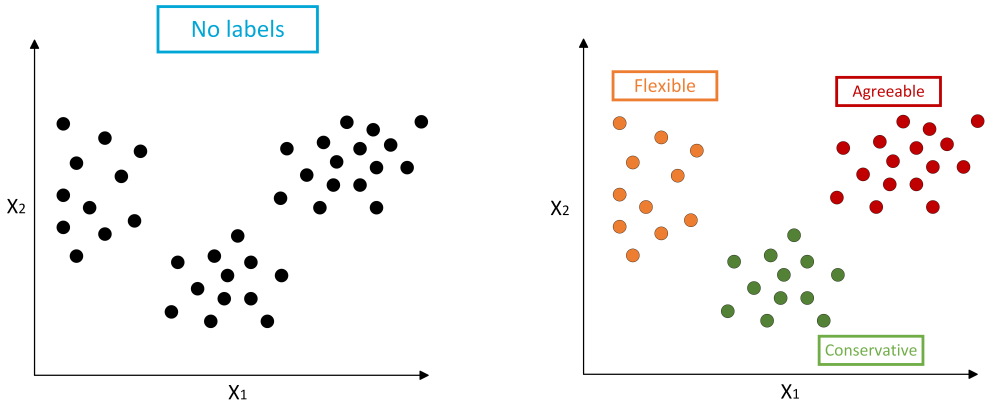


Figure 7.3: Example of clustering for customer segmentation

Supervised learning

Regression and classification are supervised learning problems. This means that the label is present, and the models aim to predict the label of a never-before-seen data point.

Figure 7.4 shows the input and output of a regression model. Data from i_1 to i_{10} are the training data, with the associated outputs. The goal is to obtain the output for a new observation (in row 11). This is a regression problem because the output is continuous.

	j1	j2	j3	j4	y
	wind speed	Temp	Time	Month	Wind Power
i1	3	20.5	01:00	1	25.5
i2	4	22.3	09:00	2	30.2
i3	4.5	25.1	08:00	4	24.1
i4	5	23.6	12:00	3	40.5
i5	6	28.8	05:00	5	39.6
i6	2	19.5	06:00	6	19.7
i7	8	18.1	11:00	7	70.2
i8	5	9	08:00	2	35.4
i9	4	5	03:00	3	36.9
i10	3.5	4	02:00	6	20.3
i11	2	7.5	20:00	10	?

Figure 7.4: Input and output data of a regression problem

7.2.2 Assessment of model accuracy

The obtained model should be able to predict the output of a never-before-seen dataset, therefore it should be generalized, and not suitable only for the training dataset. In fact, if we compute the prediction error over the training data, this does not tell us the accuracy of the prediction of an unseen point. Rather, assessing the accuracy over the training data could favor overfitting models, that follow too precisely the training dataset, including the noise. For this reason, to evaluate the performance of a model we use **testing data**.

Testing data are obtained from the initial available set of samples, dividing it into a training set and a testing set. Usually, 70% of the samples are devoted to the training set Ω and 30% to the testing set Ω^T. The model is fitted over the training set and the fitted model is assessed using it to predict the outputs for the observations in the test set. Figure 7.5 shows how the dataset previously discussed would be split.

		j1	j2	j3	j4	y
		wind speed	Temp	Time	Month	Wind Power
Training set (70%)	i1	3	20.5	01:00	1	25.5
	i2	4	22.3	09:00	2	30.2
	i3	4.5	25.1	08:00	4	24.1
	i4	5	23.6	12:00	3	40.5
	i5	6	28.8	05:00	5	39.6
	i6	2	19.5	06:00	6	19.7
	i7	8	18.1	11:00	7	70.2
Test set (30%)	i8	5	9	08:00	2	?
	i9	4	5	03:00	3	
	i10	3.5	4	02:00	6	

Figure 7.5: Split of a dataset in training set and testing set

7.3 Operational security assessment with ML models

A power system is a large interconnected network composed of generators, loads, lines, transformers and other components, all interconnected, and a fault on one of the components can have a detrimental impact on the system. Blackouts are rare, but have severe economic and societal impacts. Moreover, in case of a total blackout, the power system could require weeks to be restored. Machine learning is a great tool to predict if an outage will lead to a blackout.

Machine learning can be used to assess if the failure of a component will lead to a secure or insecure operation. This can be seen as a binary classification problem, with a high-dimensionality. Every event is described by a large number of features. For this reason, a time-domain simulation approach is relevant only for offline simulations, because of the long computational time required. Instead, a classifier can be used in real-time operations for security assessment, because it can be used close to real-time.

7.3.1 Metrics for classification

Predicted and actual security limits are different. This mismatch results in misclassification errors. The **confusion matrix**, shown in figure 7.6, can be used to summarize the predicted classes, compared to the true classes. The predictions are split into:

- TP: true positive;
- TN: true negative;
- FP: false positive;
- FN: false negative.

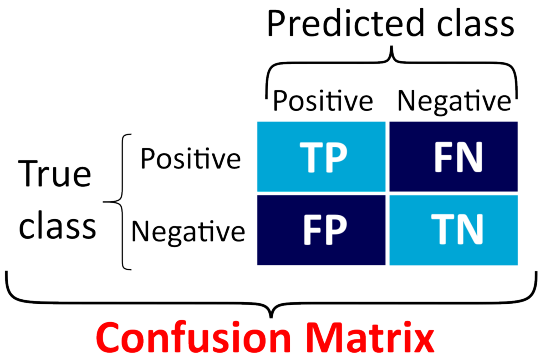


Figure 7.6: Confusion matrix

There are two types of accurate predictions, the TN and TP. Both of these are important. However, in the case of operational security assessment, TP is more important given that the consequences of not identifying an insecure data point are higher (in this example, this would lead to an unforeseen blackout). In order to take into consideration the different types of predictions, different metrics for classification are used to assess classification models' performance:

- Classification error = $\frac{N_{FP}+N_{FN}}{N_{FP}+N_{TP}+N_{FN}+N_{TN}}$
- Accuracy = $\frac{N_{TP}+N_{TN}}{N_{FP}+N_{TP}+N_{FN}+N_{TN}}$
- Precision = $\frac{N_{TP}}{N_{TP}+N_{FP}}$
- Recall = $\frac{N_{TP}}{N_{TP}+N_{FN}}$

Duality: Precision vs. Recall

Precision and recall present a duality: according to how the model is trained, it is possible to obtain either a high precision or a high recall, but not both. The decision regarding which one to favor depends on the aim of the model, therefore if TP or TN are preferable. Figure 7.7 illustrates this duality:

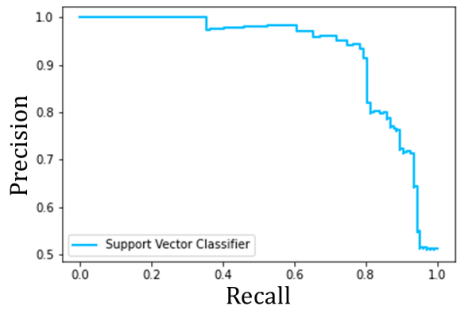


Figure 7.7: Duality between recall and precision in a two-class classification problem

The F1 score represents the harmonic mean of precision and recall and is a good way to assess the balance between the two metrics.

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{7.3}$$

To conclude, the presence of this trade-off makes it fundamental to understand the problem for which the machine learning model is being used. In fact, knowing the implications of a FP and a FN, it is possible to achieve the best possible trade-off. Going back to the power system security assessment case study, not predicting a blackout has worse consequences than wrongly predicting a blackout that does not occur. Hence, the correct forecasting of TP has to be preferred with respect to TN.

7.3.2 Fault classification

Another example of a problem than can be addressed with supervised learning is fault classification of PV modules. The aim is to detect faults among the observations. This is performed by observing the power output of the PV module and comparing it with the irradiance (solar energy density) received by the module. For example, when the irradiance remains high, but a power drop is observed, it means that a fault has occurred. This problem can be visualized in figure 7.8, where X_1 is the irradiance in W/m2 and X_2 is the electrical power output of the PV module in W.

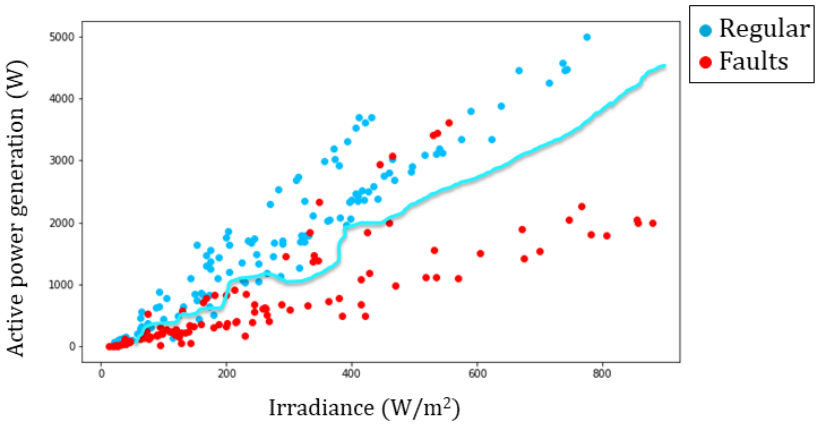


Figure 7.8: PV panel fault distribution. The light blue line represents how a fault classification model in two dimensions with SVMs would divide the regular observations from the failures.

The objective is to find a function $Y = f(X_1, X_2)$ that gives the label Y_i = -1 or +1, where -1 indicates a fault and +1 a regular observation. It is possible to address this problem with Support Vector Machines (SVM).

7.3.3 SVMs

Support vector machine is a technique that consists in creating a hyperplane (a boundary) to separate n-dimensional space into different classes. A linear hyperplane can be described by a set of parameters β as a function:

$$f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p \tag{7.4}$$

where p is the number of dimensions. In a problem with $p = 2$ (in two dimensions), the hyperplane is a straight line that divides the space in two areas, represented in red in figure 7.9, where $f(X) = 0$. The hyperplane function $f(X)$ can be used to evaluate a point in space. When $f(X) > 0$, the point is located on the upper-right side of the space and labeled as $Y_1 = 1$, while when $f(X) < 0$ it is located on the lower-left and labeled as $Y_1 = -1$. In this way, the function can be used systematically to determine to which area each point belongs and to assign it a label. Moreover, $f(X)$ gives a measure of the distance of the point from the hyperplane. In figure 7.9, there are two examples, represented by the blue dots.

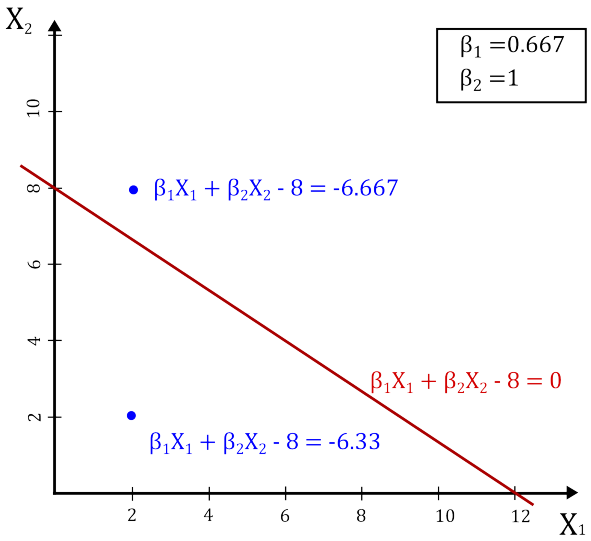


Figure 7.9: Graphic representation of a hyperplane function

Learning hyperplanes for SVMs

After seeing how a hyperplane is used to assign labels, it is necessary to understand how to find the hyperplane that has the biggest separation between the two classes. This is done in a constrained optimization problem:

$$\begin{aligned} \max_{\beta_0,\beta_1,\ldots,\beta_p} \quad & M \\ \text{subject to} \quad & \textstyle\sum_{j=1}^{p} \beta_j^2 = 1 \\ & y_i(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p) \geq M \quad \forall i \end{aligned} \tag{7.5}$$

This problem seeks to optimize the parameters β, in order to maximize the distance of the points from the hyperplane. While the first condition is necessary to normalize the parameters to 1.

The main limitation of the SVM technique is that it applies only to separable sets of data. In case the set of data is not completely separable, some points can be allowed to be located in the wrong class. The margin for which this is allowed is represented by the parameter C. The mathematical explanation of the role of C in the optimization problem is out of the scope of this lecture. Figure 7.10 illustrates how the individuated hyperplane for the same set of data changes, depending on the parameter C.

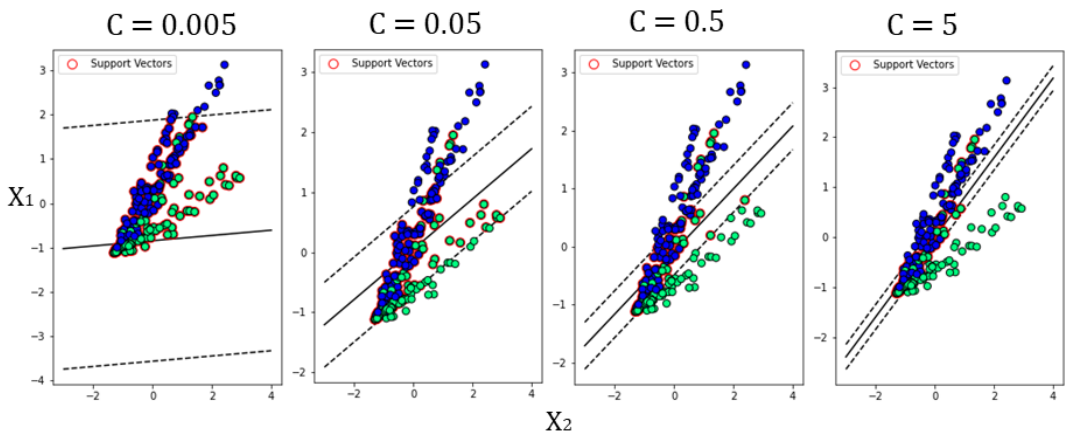


Figure 7.10: Change of the hyperplane according to different values of C

In case of non-linear hyperplanes, it is possible to use the Kernel trick to transform the problem in a linear one and then proceed as already explained.

7.4 Load and renewable generation prediction using ML models

Load and renewable generation prediction are regression problems and they are fundamental for the operation and planning of transmission and distribution networks. It is possible to predict the load of some regions with an accuracy between 1-3 %, while for other regions this value is higher, depending on the features available and the energy consumption behavior. Different models are used for low voltage and medium-high voltage feeders. In general, it is possible to observe that, when the number of feeders increases, the relative error of forecasting is decreased. This happens because the aggregated households have lower variability in consumption than a single household [4].

It is possible to diversify forecasting models depending on the forecasting horizon:

- Short-term forecast: from one hour, up to several days, with a short (5 min, 15 min) to real-time forecast resolution. It is used for short-term planning (as the balancing in the transmission system) and real-time dispatch.
- Medium-term forecast: from more than a week, up to a month. It is usually done in hours resolution. It is used for dispatch, especially of large thermal plants, that require several hours to reach the desired power output, and for long-term operational decisions.
- Long-term forecast: from one month to a year. It is done for long-term planning of generation, transmission and distribution, and for infrastructure development (for example to plan the expansion of the capacity of transmission lines).

In short-term load forecasting, many factors must be included: time factors, weather factors, extraordinary events, the type of users and their appliances. Then, the correlation between the load and the features is investigated. Figure 7.11 shows an example of negative correlation between the load and the ambient temperature, in the Netherlands. This can be explained by the fact that with higher temperatures, less heating is required. The correlation

would be different in a warmer region, where the use of air conditioning would increase the load with higher ambient temperatures.

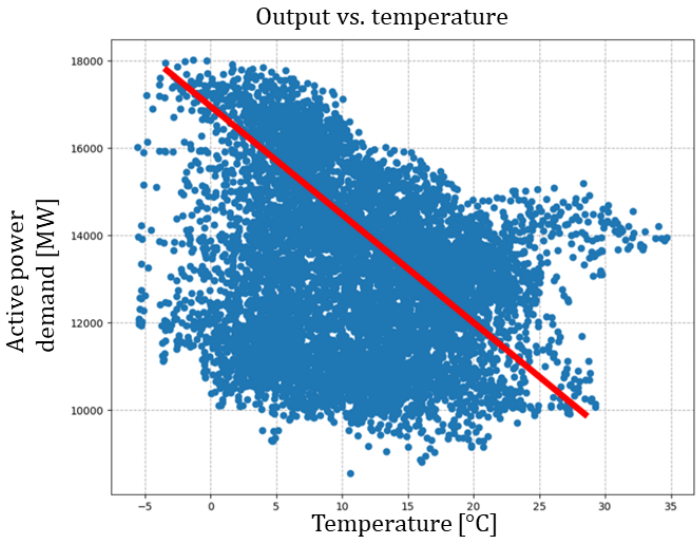


Figure 7.11: Example of negative correlation between load and ambient temperature in the Netherlands

Correlation can be also investigated between different time steps of the same feature. For example, a positive correlation has been observed between the load in one time-step and in the previous one. Similarly, a positive correlation can be observed between the consumption in one time step and the consumption at the same hour, the previous day.

For long-term forecasting, also other features must be considered, such as the age of equipment, customer behavior and population dynamics, demographic information (income, employment levels) and electricity prices.

Linear regression for load prediction

Load forecasting can be performed as a linear regression model. A general linear regression model, with a given vector of inputs $X_i^T = (x_{i1}, x_{i2}, ..., x_{ij}, ...x_{ip})$ and an output y_i, can be written as follows:

$$y_i = \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j \tag{7.6}$$

where β_j are the model coefficients or parameters, and β_0 is the intercept, or bias. The features x_{ij} can represent:

- quantitative inputs (e.g. temperature, irradiance);
- transformations of quantitative inputs, such as log, square-root or square;
- basis expansions, such as $x_j = x_1^2$, $x_j = x_2^3$, leading to a polynomial representation;
- interaction between variables, for example $x_j = x_1 \cdot x_3$.

Therefore, it is important to notice that the model is linear in the parameters β_j, no matter the sources of the variables.

The **"least square"** is a technique to individuate the parameters β_j, where these are chosen to minimize the residual sum of squares (RSS):

$$RSS(\widehat{\beta}) = \sum_{i=1}^{N}(y_i - \widehat{y_i})^2 = \sum_{i=1}^{N}(y_i - \widehat{\beta_0} - \sum_{j=1}^{p} x_{ij}\widehat{\beta_j} \tag{7.7}$$

In an ideal model, able to predict exactly the load, RSS would be equal to zero. Since RSS is a quadratic function of the parameters, its minimum always exists, but may not be unique. To evaluate the accuracy of a linear regression model, different parameters are used: R^2, the residual standard error (RSE) and the mean square error (MSE). R^2 is the most important one and measures the proportion of variability in y that can be explained using X:

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS} \tag{7.8}$$

where $TSS = \sum(y_i - \widehat{y_i})^2$ is the total sum of squares. In an ideal case, where RSS=0, R^2 would be equal to 1.

Shrinkage methods

The main drawback of linear regression models is that there can be different units of measure within the same problem, hence different scales. As a consequence, the parameters can reach really large values. To solve this problem, shrinkage methods are used to find the smallest possible parameters. This process is called "regularization". There are two main shrinkage methods: Ridge Regression and Lasso Regression.

SVMs for regression

Support vector machines can be also used for regression problems. In this case, the parameters of the hyperplane are estimated in order to fit the largest possible number of points within the region surrounding the hyperplane. This region is known as "margin". It is possible to estimate these parameters through an optimization problem. Figure 7.12 gives a visual representations of the hyperplane and the margin. With the Kernel trick it is possible to transform the data to make them suitable to fit a linear hyperplane.

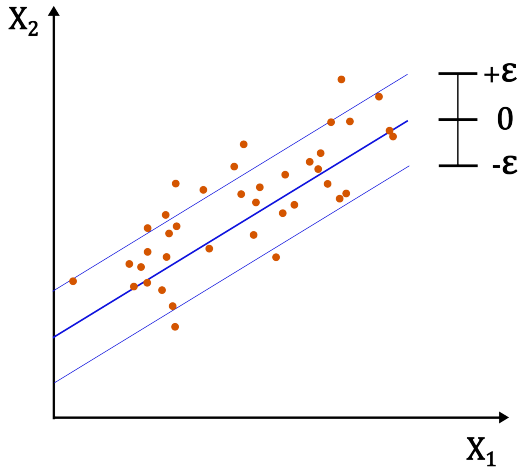
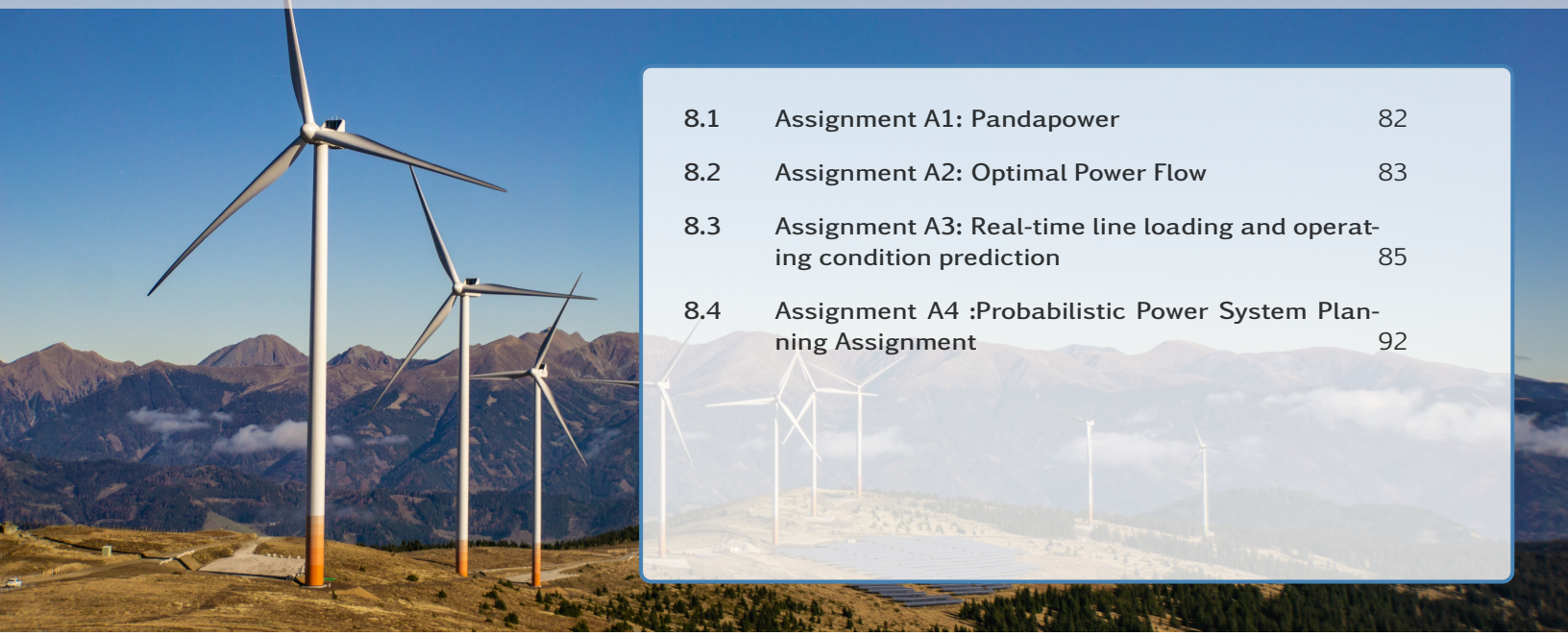


Figure 7.12: Graphic representation of a hyperplane used for regression

One of the main advantages of SVMs is that they are robust to outliers, in the sense that it is not important how far the points are from the margin. Additionally, it has excellent generalization capability, with high prediction accuracy, and is easy to implement. The disadvantages are that SVMs are not suitable for large datasets and that they do not perform well when the dataset is not separable. Moreover, it is not possible to know in advance if a dataset is separable, therefore the only solution is to implement SVMs and assess the results.

8. Assignments



8.1 Assignment A1: Pandapower

8.1.1 Objective

For modeling and simulating Smart Grids, all assignments rely on the same simulation tool, which is pandapower [**Pandapower2021**]. The tool provides advantages of being open source software and cover different scopes of grid simulations. It is a Python based library to perform static power flow simulations. To work with pandapower, we will implement or provide scripts, which are written in Python, to perform a pandapower based simulation. For all Python scrips, we are using Spyder as the integrated development environment (IDE) and Anaconda.

Besides verifying a successful installation, the overall learning objectives of the first assignment are as follows:

1. Learn about pandapower [**Pandapower2021**]

2. Perform a static power flow simulation

8.1.2 Methodology

- Download/install Anaconda `https://www.anaconda.com/distribution/`
- Install pandapower (http://www.pandapower.org/start/). Use *Anaconda prompt* instead of the default *cmd* to run the "pip install pandapower" command.
- Pandapower manual can be found at `https://pandapower.readthedocs.io/`.
- Download Assignment zip file (Brightspace), and unzip it.
- Start Spyder
  - Open the file case4gs_simple.py from assignments folder
  - Execute the file
  - Reduce the load active power (connected to bus 2, in pandapower bus index starting from 0) from 200 MW to 80 MW in the case file and observe the changes in powerflow (Use pandapower documentation).

8.1.3 Report

Write a 2-page report that covers

- all observations made during the above instructions,
- a 1-line (a.k.a. single-line) diagram of the pandapower case4gs network and

Hint: Plotting a 1-line diagram with pandapower is possible, but the visualisation of each component is very simplified. Instead, you can use Paint, PowerPoint or other tools to draw the grid by your own.

Please consider the assessment matrix given in the introduction while doing your reports and be aware that exceeding the page limit lowers the report's grade.

8.2 Assignment A2: Optimal Power Flow

8.2.1 Objective

This assignment focuses on pandapower applications for optimal power flow and probabilistic power flow calculations. A modified version of the IEEE 9-bus system (to include a wind power plant) is used as case study. This part of the lecture has tree learning objectives:

1. Describe the rationale behind optimal power flow (OPF) calculation.
2. Describe the rationale behind probabilistic power flow (PPF) calculation.
3. Evaluate power system steady-state performance based OPF and PPF

By performing calculations in pandapower, and analyzing the numerical outcomes (bus voltages, branch currents, etc.), this assignment will help you consolidate the concepts learnt during the lecture.

8.2.2 Methodology

Read the pandapower manual carefully to learn how to do power flow and optimal power flow calculations. In Brightspace, you can find the following python files in assignment 2 folder:

1. opf_ieee9_wind.py - The model of the modified IEEE 9-bus test system (baseline for Task 1).
2. prob_opf_ieee9_wind.py - Example to run PPF (baseline case for Task 2).

Task 1

Open the file opf_ieee9_wind.py in Spyder. This is the base case. You can run this file to check working of baseline case.

- Run economic dispatch for the modified IEEE 9-bus system by considering different locations (bus 7 or bus 5) of the wind power plant. TIP: You will need to change the index of the transformer that connects the wind power plant system to the main 9 bus system. Refer the one line diagram.
- For each location of the wind power plant, increase the demand in each node by 50% or by 100%, and run economic dispatch again.
- Reflect on the results from each economic dispatch, w.r.t. baseline case, by analyzing the change in voltage magnitudes and the branch loading

Task 2

Open the file prob_opf_ieee9_wind.py in Spyder. This is the base case (N=100). You can run this file to check working of baseline case.

- Run PPF for the modified IEEE 9-bus system by considering different locations (bus 7 or bus 5) of the wind power plant.
- For each location of the wind power plant, set the reactive power (net.sgen.q_mvar) (corresponding to 0.95 (overexcited), or a fixed value of 0.95 (underexcited) as power factor) of the wind power, and run economic PPF again.
- Reflect on the results from each PPF, w.r.t. baseline case, by analyzing the change in variability of voltage magnitudes and the branch loading.

Note: take into account that not every sample operating condition (associated to random sampled values of load demand and wind power output) leads to a convergent OPF.

8.2.3 Report

Prepare a short report (max. 4 A4 pages, free format) showing a summary of the results about the system performance (i.e. results from all OPF and PPF calculations). For both tasks, create one figure for the variability of voltage magnitude of all buses (including a depiction of the upper and lower bus voltage magnitude limits), another one for variability of the loading of all branches, and another one for variability of active power losses in all branches. Figures of different cases can be merged into a single to better illustrate the effects

of different changes in the system, e.g. effect of different wind power plant location on bus voltage profile. You can also edit the scripts to generate figures to depict only the highest and lowest value of a power flow outcome (e.g. bus voltage magnitude).

Write some text (max half a page) to reflect on the results from each probabilistic power flow calculation w.r.t. baseline case of Task 2 (i.e. running PPF by using the provided files with original settings), by analysing the change in the voltage magnitudes and the branch loading. Explain, from a sensitivity analysis point of view, the effect of different locations and different reactive power set-points of the wind power plant on the variability of voltage profiles, branch power flows, and active power losses of the modified version of the IEEE 9-bus system. Example point of reflection: The short electrical distance between the wind power plant and the largest load, and the high reactive power output associated with the given XX power factor resulted in an excess of reactive power at bus YY and the nearby buses ZZ. This increased the voltage magnitude on one hand, and the power losses due to an increased current flowing through the adjacent transmission branches.

8.3 Assignment A3: Real-time line loading and operating condition prediction

8.3.1 Objective

Due to the increasing levels of distributed generation (e.g. PV systems, wind turbines), predicting power fluctuations in power systems is getting more challenging. To address this, accurate security measures and frequent monitoring are essential. One of those security measures includes monitoring the loading percentage of lines to detect when they reach dangerous values which could later cause line failure. Applying machine learning to estimate whether some operating conditions could lead to a dangerous line loading percentage on a specific bus could be beneficial in terms of both detection and adaptability. This assignment is focused on utilizing machine learning models for the accurate prediction of secure and insecure operating conditions measured as line loading percentages.

The objectives of this assignment is to use supervised learning (i.e., classification and regression) for power systems operational tasks. You will develop a regression model to perform real-time line loading prediction and a classification model to perform overload security assessment. Figure 8.1 shows an example of the expected outcome from this assignment. The x-axis represents the time (in minutes) as measurements were recorded, while the y-axis represents a line loading percentage (in %). The orange line represents the actual line loading percentage data. The first goal of the assignment is to predict security in the classification problem. There, the red line illustrates the loading percentage threshold which would distinguish a secure from an insecure line loading operation. The second goal of the assignment is to predict the line loading values illustrated with the blue line which is the output of the regression model.

Software tools and packages

1. Pandapower: An open source tool for power system modeling, analysis and optimization with a high degree of automation.

2. Sckitlearn (Python): A library for machine learning development in Python. It provides an easy-to-use interface for efficient machine learning models development for

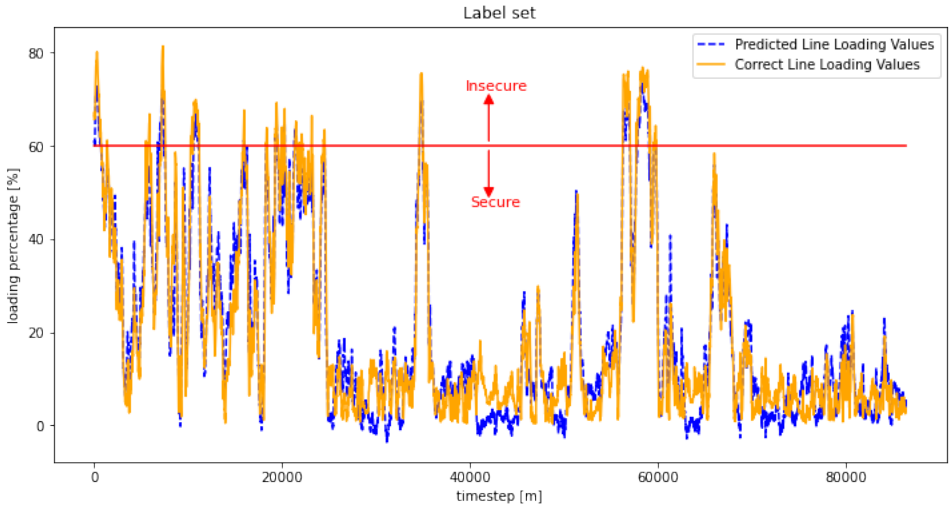


Figure 8.1: Example of this assignment tasks and the expected outcome: prediction of a line loading percentage and its operating condition (secure or insecure).

classification, regression, clustering, and dimensionality reduction tasks.

3. Matplotlib: A Python library for creating static, animated, and interactive visualizations.

4. Simbench: A cross-platform benchmarking framework and can be re-targeted to new architectures with minimal effort.

5. Numpy: A fundamental package for scientific computing with Python.

6. Pandas: A fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

7. Jupyter Notebooks: Web-based Python interface that allow rapid Python code development and debugging.

Assignment Submission

This assignment includes 8 tasks with several questions. Your task is to answer each question in a Jupyter notebook and summarise the findings in a report. You will submit the report to Brightspace as a PDF document.

8.3.2 Methodology

The methodology of this assignment concerns dataset information extraction, time series calculation, loading of simulation results in machine learning model, data-prepossessing, regression and classification.

Dataset Information Extraction

Initially, you will use the Simbench library to obtain a network with load and generation profiles for 1 year in 15 minute time-steps. The network includes 82 buses, 79 loads, 98

static generators, 113 lines, 3 transformers, 14 substations. You will only keep the datapoints from the first 2 months for computational complexity reduction purposes.

Running the Timeseries Calculation for the Line Loading Percentage Calculations

To evaluate the loading percentages of the lines, timeseries calculations are executed by solving powerflows based on the profiles imported using the Simbench library. Accordingly, the load and generation profiles are used to create 3 types of constant controllers where the corresponding values are followed by the power flow solvers. These controllers are:

1. Static generator active powers
2. Load active powers
3. Load reactive powers

The timeseries calculations are executed using the Pandapower library.

Loading of simulation results and generation of machine learning model datasets

In this part you will load the simulation results which were saved in the previous section and load the features which will be used by the machine learning model from the Simbench profiles.

The features include network's static generators active powers [MW], and load active [MW] and reactive powers [MVAr]. However, the dataset does not include all the electrical variables for all nodes and lines as the network operator does not have full observability of the network. As you can see, the data does only include around 10% of the possible features. Hence, the dataset features are:

$$X = [P_0^{\text{sgen}}, P_1^{\text{sgen}}, ...P_{n_g}^{\text{sgen}}, P_0^{\text{load}}, P_1^{\text{load}}, ...P_{n_l}^{\text{load}}, Q_0^{\text{load}}, Q_1^{\text{load}}, ...Q_{n_l}^{\text{load}}], \quad \in \mathcal{R}^{T\times(2n_l+n_g)} \tag{8.1}$$

where T is the number of timesteps within the dataset, n_l is the number of observable loads within the network, n_g is the number of observable static generators within the network, P_i^{sgen} is the active power of static generator i, P_i^{load} is the active power of load i ,Q_i^{load} is the reactive power of load i.

In this part, the data will be split between the training and testing sets. The first set will be used for the training phase of the models, where the parameters or coefficients will be evolved to allow better prediction. On the other hand, the test set will only be used after the model is trained, to evaluate the generalization abilities of each model. In other words, the test set will allow us to have a better idea whether the model would perform well when given new data. The dataset is split in 2 parts with the training set being 70% of the data X with their targets y, and the test set being the rest 30% of those sets. The datapoints are distributed randomly to the two sets.

Data preprocessing

Data prepocessing is a step done before training machine learning models.

Pre-processing cleans the data, and for instance, can transform the data into a format that can be considered by the machine learning models.

Standardization can help when there are large differences between the value ranges of the dataset features, or simply when they are measured in different measurement units. These differences in the ranges of initial features reduce the accuracy of machine learning models. As an example, if we have 2 dataset features one for a static generator's power output in Watt [W] and another feature for the bus voltage in per unit [p.u.], the model could be quite sensitive or focused on the generator power feature, simply because of the scaling rather than the importance of that feature. Applying standardization performs:

$$\tilde{x}_i^j = \frac{x_i^j - \hat{x}^j}{\sigma_x^j}, \quad \forall i \in [1, ..., T], \quad \forall j \in [1, n_j] \tag{8.2}$$

where $\tilde{x}_i^j$ is the i^{th} timestep's, j^{th} feature's standardized input, x_i^j is the i^{th} timestep's, j^{th} feature's input, $\hat{x}^j$ is the mean value of the j^{th} feature, and σ_x^j is the j^{th} feature's standard deviation. As an example, an input dataset with values $[[0, 2], [1, 4]]$, when standardized will become $[[-1, -1], [1, 1]]$. Therefore, it is suggested that standardization is performed almost always before training on a dataset.

To standardize generally means changing the values so that the distribution's standard deviation equals one. StandardScaler results in a distribution with a standard deviation equal to 1. The variance is equal to 1 also, because variance is equal to the standard deviation squared ($Var = \sigma^2$ and $1^2 = 1$). StandardScaler makes the mean of the distribution 0. About 68% of the values will lie be between -1 and 1.

Linear Regression

Linear Regression fits a linear model with parameters $W = (w_1, \ldots, w_p)$ to minimize the residual sum of squares between the observed targets (y) in the dataset, and the targets predicted by the linear approximation ($f(x)$). The equation of a linear regression prediction is:

$$f(\tilde{x}_i) = W\tilde{x}_i + w_0, \quad \in \mathcal{R} \tag{8.3}$$

where $\tilde{x}_i$ is the i^th datapoint of the standardized input, w_0 is the intercept, and W are the model's parameters. The training phase of the model will try to update the initial W, and w_0 values such that the model outputs will be as close as possible to the actual labels y.

At this point, the first regression model is initialized, trained and evaluated using the R^2 score. This score measures the proportion of variability in the target that can be explained using the input, and is computed using:

$$R^2 = 1 - \frac{RSS}{TSS} \tag{8.4}$$

where:

$$RSS = \Sigma_{i=1}^{N} (y_i - f(\tilde{x}_i))^2 \tag{8.5}$$

$$TSS = \Sigma_{i=1}^{N} (y_i - \hat{y})^2 \tag{8.6}$$

with $\hat{y}$ being the mean value of the output y. Given that the R^2 score can showcase the proportion of the line loading percentage variance that can be explained by our input features (i.e., static generators active powers, load active and reactive powers) through our trained

model, it is a good measure as to evaluate whether the model can accurately predict the loading percentage given our inputs.

The training and test results are shown both through the R^2 score and a graph which shows the actual values vs the predicted values.

Classification

The line operating condition in the context of these lecture will be considered as within dangerous limits when its loading percentage is equal or higher than 60%. Hence, the target label of each datapoint needs to be determined. Loading percentages higher than 60% will be labeled as 1, while the rest of the line loading percentages will be labeled as 0.

Classification 1: Linearized Logistic Regression

This class implements regularized logistic regression using the 'liblinear' library, 'newton-cg', 'sag', 'saga' and 'lbfgs' solvers. Note that regularization is applied by default. It can handle both dense and sparse input. Use C-ordered arrays or CSR matrices containing 64-bit floats for optimal performance; any other input format will be converted (and copied)

8.3.3 Assignment and Report

Task 1 relates to dataset information extraction, while Tasks 2 and 3 relate to the machine learning models development for line loading and operating condition prediction. Finally, Tasks 4 and 5 relate to regression, and tasks 6, 7, & 8 to classification.

Task 1: Extract Generic Information

1. Find the maximum active power consumption per load within the load profile (per column)

2. Find the maximum active power generation from the static generators profile

3. Find the maximum apparent power consumption magnitude from the load no.0 active and reactive power consumption profiles (Hint: Load no.0 is the 1st Column of the load_p and load_q dataframes). In your report, also include the timestep index of the found maximum apparent power.

Task 2: Regression Task Targets Creation

You should address the following questions in both the Jupyter notebook and your report.

1. Find the highest line loading percentage within the dataset

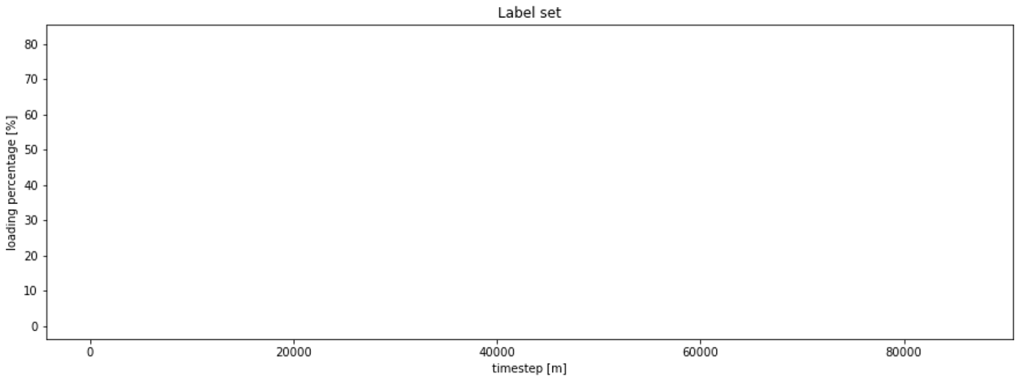
2. Find the line ("line_y") that has the highest loading percentage within the dataset

3. Save all loading percentage values of line "line_y" as the vector 'y'. The selected line will be used as the dataset targets to be predicted by the regression problems.

Task 3: Matplotlib Introduction

You should address the following questions in both the Jupyter notebook and your report.

1. Plot the data labels in a line plot with x-axis as the minutes past compared to the first datapoint (remember: the dataset was created in 15 minute intervals). Add axis labels and appropriate title to the figure. The shape of the figure should have 15 inch width and 5 inch height. An example of the figure size and axes labels is shown below.



Task 4: Tuning of SVR hyperparameter

Support vector regression is a more complex task with more hyperparameter options compared to linear regression. The coding implementation is quite similar to the linear regression model. SVR hyperparameters include:

- *kernel*: Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to precompute the kernel matrix
- *degree*: Degree of the polynomial kernel function ('poly'). Ignored by all other kernels
- *gamma*: Kernel coefficient for 'rbf', 'poly' and 'sigmoid'
- *C*: Regularization parameter. The strength of the regularization is inversely proportional to C

1. Change the hyperparameter 'C' between the values 0.1 and 5 in order to improve the training error. Report the resulted training and test errors for 3 different 'C' values. Based on the training and test error results, which model would you select and why?

A sample answer is given below:

"A. For the hyperparameter 'C' = ___ the resulted training and test errors are _ and _"
"B. For the hyperparameter 'C' = ___ the resulted training and test errors are _ and _"
"C. For the hyperparameter 'C' = ___ the resulted training and test errors are _ and _"
"My chosen hyperparameter 'C would be _____, because _____"

Task 5: Tuning of Decision Tree Regression hyperparameter

The coding implementation is quite similar to the previous ones, in addition to the larger amount of hyper-parameters which can be given as inputs when initializing the model compared to the linear regression model.

- *min_samples_split*: The minimum number of samples required to split an internal node
- *min_samples_leaf*: The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least ”min_samples_leaf” training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression
- *min_weight_fraction_leaf*: The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node.
- *max_features*: The number of features to consider when looking for the best split

1. Change the hyperparameter ’min_samples_split’ to values between 2 and 50 to improve the current model (if you are not able to improve the model explain why). Report the resulted training and testing errors for the models with 3 different hyperparameter values from these limits. Based on the resulted training and test errors, which model would you select and why?

A sample answer is given below:
”A. For the hyperparameter ’min_samples_split’ = ___ the resulted training and test errors are _ and _”
”B. For the hyperparameter ’min_samples_split’ = ___ the resulted training and test errors are _ and _”
”C. For the hyperparameter ’min_samples_split’ = ___ the resulted training and test errors are _ and _”
”My chosen hyperparameter ’min_samples_split’ would be _____, because _____”

Task 6: Creation of Classification Label Set

1. Using the y_train set, create the classification labels vector y_train_class with values 0 and 1 as explained in the classification section (loading percentages higher than 60% will be labeled as 1, while the rest of the line loading percentages will be labeled as 0. Print the ratio of training set safe datapoints with respect to all the training set data points.)
2. Using the y_test set, create the classification labels vector y_test_class with values 0 and 1 as explained in the classification section (loading percentages higher than 60% will be labeled as 1, while the rest of the line loading percentages will be labeled as 0. Print the ratio of test set safe datapoints with respect to all the test set data points)

Task 7: Develop an SVM Classification Model

1. Develop an SVM classifier using scikit learn
2. Train the SVM classifier (Hint: Similar to Classifier 1)

3. Report training & test scores
4. Plot the classification results for the test set using the plot_classification_prediction() function
5. Plot the confusion matrix for the test set

Optional - Task 8: Develop a Decision Tree Classification Model

1. Develop a Decision Tree classifier using scikit learn
2. Train the Decision Tree classifier (Hint: Similar to Classifier 1)
3. Report training & test scores
4. Plot the classification results for the test set using the plot_classification_prediction() function
5. Plot the confusion matrix for the test set

Report

Answer each question of each task in a Jupyter notebook and summarise the findings (results, graphs, short discussion) in a max 3-page PDF report.

8.4 Assignment A4 :Probabilistic Power System Planning Assignment

8.4.1 Objective

Modern power distribution systems are increasingly affected by uncertainty due to distributed generation, renewable integration, and evolving load patterns. As a result, traditional deterministic planning approaches are often insufficient to capture rare but critical operating conditions.

The objective of this assignment is to introduce a **probabilistic power system planning workflow** using **generative statistical models**. In particular, students will explore how **high-dimensional Gaussian models** and **Gaussian Mixture Models (GMMs)** can be used to generate synthetic load scenarios and assess voltage behavior under uncertainty.

Using the IEEE 33-bus distribution system as a case study, this assignment demonstrates how probabilistic simulations can reveal voltage variability and potential risk areas that may not be visible in deterministic analyses.

8.4.2 Assignment Structure

This assignment consists of three main parts:

- **Part I: Reading and Understanding** (no coding required)
- **Part II: Coding Tasks** (Tasks 1–3)
- **Part III: Analysis Task** (Task 4)

Students are expected to carefully read the explanatory sections of the notebook before starting the coding tasks.

8.4.3 Part I: Reading and Understanding

Before writing any code, students should read and understand the following concepts presented in the notebook:

- The motivation for probabilistic power system planning and its advantages over deterministic approaches.
- Modeling bus-level active and reactive power demands using **multivariate Gaussian distributions**.
- Visualization of load variability in a two-dimensional (P, Q) space.
- The role of **Gaussian Mixture Models (GMMs)** in capturing multimodal load behavior.
- Conditional sampling from a trained GMM to generate scenarios with specified average load levels.

Understanding these ideas is essential for correctly completing the coding and analysis tasks.

8.4.4 Part II: Coding Tasks

Task 1: Sampling Load Data from a Conditional GMM

The goal of this task is to generate synthetic load scenarios using a trained Gaussian Mixture Model.

- Write the code to call the function `sample_from_conditional_gmm(...)` **twice**:
  - once for a **lower mean load** scenario,
  - once for a **higher mean load** scenario.
- The function returns synthetic active power values (in MW) for each bus in each scenario.
- Take the absolute value of the sampled loads to ensure non-negative power consumption.

Task 2: Completing the Voltage Computation Function

In this task, you will complete the helper function `run_dc_scenarios_and_get_angles_and_voltages(...)`.

For each load scenario, the function should perform the following steps:

1. Create a fresh pandapower IEEE 33-bus network.
2. Remove any pre-existing loads from the network.
3. Add new loads for all buses using the synthetic load scenario.
4. Run a power flow calculation.

The function should return the voltage results for all scenarios.

Task 3: Voltage Scenario Generation and Visualization

Using the function developed in Task 2:

- Run the power flow for all generated scenarios.
- Extract the bus voltage magnitudes.
- Visualize the voltage distributions using boxplots.

The resulting plots should illustrate how voltage behavior changes across different scenarios and load levels.

8.4.5 Part III: Analysis Task

Task 4: Analysis of Probabilistic Power Flow Simulations

Based on the lecture content and your simulation results, answer the following questions:

1. What is the added value of probabilistic power flow analysis compared to a deterministic simulation?
2. How can this added value be exploited when planning modern power distribution systems?
3. So far, the analysis has focused on technical aspects. How can probabilistic power flow simulations also support financial analysis in distribution system planning?

Your answers should be concise and refer to the observed voltage distributions and scenario-based results.

8.4.6 Report

Prepare a short report (maximum **4 A4 pages**, free format) summarizing the results and conclusion obtained in this assignment.

Bibliography

[1] Mahamad Nabab Alam. *State-of-the-Art Economic Load Dispatch of Power Systems Using Particle Swarm Optimization*. 2018. DOI: 10.48550/ARXIV.1812.11610. URL: https://arxiv.org/abs/1812.11610.

[2] *An introduction to statistical learning*. Springer Texts in Statistics. Springer New York, NY, 2013. ISBN: 978-1-4614-7138-7. URL: https://doi.org/10.1007/978-1-4614-7138-7.

[3] F. M. Gonzalez-Longatt et al. "Identification of Gaussian mixture model using Mean Variance Mapping Optimization: Venezuelan case". In: *2012 3rd IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*. 2012, pp. 1–6. DOI: 10.1109/ISGTEurope.2012.6465672.

[4] Stephen Haben et al. "Review of low voltage load forecasting: Methods, applications, and recommendations". In: *Applied Energy* 304 (Dec. 2021), p. 117798. DOI: 10.1016/j.apenergy.2021.117798. URL: http://dx.doi.org/10.1016/j.apenergy.2021.117798.

[5] Hoan Pham and José Rueda. "Probabilistic evaluation of voltage and reactive power control methods of wind generators in distribution networks". In: *IET Renewable Power Generation* 9 (Apr. 2015), pp. 195–206. DOI: 10.1049/iet-rpg.2014.0028.

[6] JosÉ L. Rueda, Delia G. Colome, and Istvan Erlich. "Assessment and Enhancement of Small Signal Stability Considering Uncertainties". In: *IEEE Transactions on Power Systems* 24.1 (2009), pp. 198–207. DOI: 10.1109/TPWRS.2008.2009428.

[7] José L. Rueda and István Erlich. "Optimal dispatch of reactive power sources by using MVMO¡sup¿s¡/sup¿ optimization". In: *2013 IEEE Computational Intelligence Applications in Smart Grid (CIASG)*. 2013, pp. 29–36. DOI: 10.1109/CIASG.2013.6611495.

[8] scikit-learn. *Glossary*. Accessed on 18-09-2022. URL: https://scikit-learn.org/stable/glossary.html#term-imputation.

[9] Selin Yilmaz et al. "Analysis of demand-side response preferences regarding electricity tariffs and direct load control: Key findings from a Swiss survey". In: *Energy* 212 (Dec. 2020), p. 118712. DOI: 10.1016/j.energy.2020.118712. URL: http://dx.doi.org/10.1016/j.energy.2020.118712.

A. Appendix

A.1 Crew

- Responsible lecturer
  - Peter Palensky
  - Dept. Electric Sustainable Energy / EWI
  - http://ese.ewi.tudelft.nl
  - P.Palensky@tudelft.nl
- Other instruction team
  - Jose Rueda J.L.RuedaTorres@tudelft.nl (lectures)
  - Jochen Cremer J.L.Cremer@tudelft.nl (lectures)
  - Pedro Vergara Barrios P.P.VergaraBarrios@tudelft.nl (lectures)
  - Jan Ot Pina Urgell J.O.PINAURGELL@tudelft.nl (assignments)
  - Weijie Xia W.Xia@tudelft.nl (assignments)
- Contact hours: Announced at end of every lecture

A.2 Learning Objectives

After this course students are able to:

- describe the smart functions of energy management systems,
- deploy software-based models for steady-state and market performance analysis,
- synthesize operational scenarios for insecurity risk identification,

- apply data analytics and optimization for intelligent flexibility management, and
- solve operational security threats of systems (e.g. TSO-DSO, offshore/onshore systems, large/small size hubs) with different characteristics.

A.3 Lectures

The lecture consists of the following lecture units:

1. Digital energy (**Peter**)
2. Technical performance: Assessment of steady-state security (**Jose**)
3. Optimal power flow-based techno-economic assessment (**Jose**)
4. Probabilistic evaluation of system performance (**Jose**)
5. Energy management (**Peter**)
6. Data-driven distribution systems planning (**Pedro**)
7. Machine learning supported real-time system management (**Jochen**)

A.4 Assignments

All lecture units have Assignments to be done at home, to be done on groups of three.
Please enroll and submit (NO resubmission/iteration!) via Brightspace.
Assignment reports (PDF) are due 24h before the next lecture. Title page should contain:

- SET3065 Intelligent Electrical Power Grids
- Year
- Assignment Number / Assignment Title
- Group Number
- Name 1 (Number 1)
- Name 2 (Number 2)
- Name 3 (Number 3)

Page numbers is a must-have. Figures need a title and a number. All axes of a graph need a unit and a meaning (e.g., MW and P).

A.5 Literature

Only if you want to get additional details and insight, the lecture material is sufficient for the exam. No need to buy (read it in the library, most free online when accessed from within EWI network).

- Ch. 6 of Book by Pieter Schavemaker, and Lou van der Sluis, "ELECTRICAL POWER SYSTEM ESSENTIALS", John Wiley and Sons, 2009, ISBN: 978-0470-51027-8 (for Lecture 1)
- Palensky et al, "Cosimulation of Intelligent Power Systems: Fundamentals, Software Architecture, Numerics, and Coupling", IEEE Industrial Electronics Magazine, Vol. 11 (1) , pp. 34-50 (for Lecture 1)
- Palensky et al, "Applied Cosimulation of intelligent power systems: Implementation, usage, and examples", IEEE Industrial Electronics Magazine, Vol. 11 (2) (for Lecture 1)
- Synchronized Phasor Measurements and Their Applications, A. G. Phadke and J. S. Thorp, Springer (for Lecture 1)
- Monti, A.; Muscas, C. and Ponci, F.,"Phasor measurement units and wide area monitoring systems", Academic Press, 2016, ISBN: 978-0-12-804569-5, free download via https://www.sciencedirect.com/science/book/9780128045695 (for Lecture 1)
- Terzija, V.; Valverde, G.; Cai, D.; Regulski, P.; Madani, V.; Fitch, J.; Skok, S.; Begovic, M. M. and Phadke, A.,"Wide-area monitoring, protection, and control of future electric power networks," Proceedings of the IEEE, IEEE, 2011, 99, 80-93 (for Lecture 1)
- Chapter 1, 2, and 5 of A. Abur and A. G. Exposito, "Power system state estimation: theory and implementation," CRC press, 2004. ISBN: 0-8247-5570-7 (for Lecture 2)
- Ch. 4 and appendix D of Ray D. Zimmerman Carlos E. Murillo-Sanchez, " Matpower 6.0: User's Manual," December 16, 2016. http://www.pserc.cornell.edu/matpower/ (for Lecture 2)
- Ch. 6 and 7 and appendix D of Ray D. Zimmerman Carlos E. Murillo-Sanchez, " Matpower 6.0: User's Manual," December 16, 2016. http://www.pserc.cornell.edu/matpower/ (for Lecture 2)
- C.-M. Huang, S.-J. Chen, Y.-C. Huang, and H.-T. Yang, "Comparative study of evolutionary computation methods for active-reactive power dispatch," IET Generation, Transmission & Distribution, vol.6, no.7, pp. 636- 645, July 2012. PAPER CAN BE DONWLOADED FOR FREE if students are connected to EWI network. (for Lecture 3)
- Mashwani, Wali & Ruqayya, Ruqayya & Brahim Belhaouari, Samir. (2021). A Multi-swarm Intelligence Algorithm for Expensive Bound Constrained Optimization Problems. Complexity. 2021. 1-18. 10.1155/2021/5521951 (for Lecture 3)
- Vaishali Sohoni, S. C. Gupta, and R. K. Nema, "A Critical Review on Wind Turbine Power Curve Modelling Techniques and Their Applications in Wind Based Energy Systems", Journal of Energy, vol. 2016, Article ID 8519785, pp. 1-19, June 2016. (for Lecture 4)

- W. Xia, P. P. Vergara, et. at. “Comparative Assessment of Generative Models for Transformer- and Consumer-Level Load Profiles Generation”, Sustainable Energy, Grids, and Networks, 2024. (for Lecture 6)
- Chai, S., Chadney, G., Avery, C., Grunewald, P., Van Hentenryck, P., & Donti, P. L. (2024). Defining’Good’: Evaluation Framework for Synthetic Smart Meter Data. arXiv preprint arXiv:2407.11785. (for Lecture 6)
- Ardizzone, Lynton, Carsten Lüth, Jakob Kruse, Carsten Rother, and Ullrich Köthe. ”Guided image generation with conditional invertible neural networks.” arXiv preprint arXiv:1907.02392 (2019). (for Lecture 6)
- Lin, Nan, Peter Palensky, and Pedro P. Vergara. ”EnergyDiff: Universal Time-Series Energy Data Generation using Diffusion Models.” arXiv preprint arXiv:2407.13538 (2024). (for Lecture 6)
- Duque, Edgar Mauricio Salazar, Juan S. Giraldo, Pedro P. Vergara, Phuong H. Nguyen, Anne van der Molen, and J. G. Slootweg. ”Risk-aware operating regions for PV-rich distribution networks considering irradiance variability.” IEEE Transactions on Sustainable Energy 14, no. 4 (2023): 2092-2108. (for Lecture 6)
- Yilmaz, Selin, Xiaojing Xu, Daniel Cabrera, Cédric Chanez, Peter Cuony, and Martin K. Patel. ”Analysis of demand-side response preferences regarding electricity tariffs and direct load control: Key findings from a Swiss survey.” Energy 212 (2020): 118712. (for Lecture 7)
- Haben, Stephen, Siddharth Arora, Georgios Giasemidis, Marcus Voss, and Danica Vukadinović Greetham. ”Review of low voltage load forecasting: Methods, applications, and recommendations.” Applied Energy 304 (2021): 117798. (for Lecture 7)
- Tawn, Rosemary, and Jethro Browell. ”A review of very short-term wind and solar power forecasting.” Renewable and Sustainable Energy Reviews 153 (2022): 111758. (for Lecture 7)

A.6 Time budget

4 ECTS = 4x28h = 112h

- 16h (8x2h) lecture/lab
- 82h Assignments
- 14h preparation exam

A.7 Assessment

The grade is formed based on the following elements:

- 50% Assignment reports (in time!)

LO/Level	Knowledge	Insight	Application	Evaluation	Total
LO1	10%/2q	10%/2q		5%/1q	25%
LO2	20%/4q	10%/2q			30%
LO3	5%/1q		5%/1q	10%/2q	20%
LO4	5%/1q		10%/2q	10%/2q	25%
Total	40%/8q	20%/4q	15%/3q	25%/5q	100%

Table A.1: Exam Assessment Matrix (percent of grade, number of questions.)

  - Figures/tables and text/equations get grade of 0-5 each (see A.3) → 0-10 points for each assignment.
  - Sum of all assignment points is divided by 8 → 0-10 points in total
  - Report submission not in time? → 0 points.
  - Insights derived and relevant figures supporting analysis (quality of figures is important, use matplotlib (python) or MATLAB, rather than screenshots).

- 50% multiple choice MapleTA Computer Exam
  - Questions are of theoretical nature or simple calculations.
  - Approximately 30 questions, 120min
  - Paper and calculator allowed (but not needed)
  - 0-10 points in total

- No minimal grade for each of the two parts.
- Max final grade is 10, you need >= 6 to pass (in total, no threshold for individual elements).
- All assignments and their reports are mandatory.
- Grade rounded to .0 or .5 (e.g., 5.8 → 6.0)

Grade	Quality of figures and tables	Quality of text and equations
0	IF IN ALL FIGURES/TABLES: Labels/captions/legend missing. Illegible text. Overcrowding. Unidentifiable elements (e.g. curves). Poor resolution.	Incomplete. Document contains serious spelling and grammatical errors. Report with more than one page over-length.
1	IF IN AT LEAST TWO FIGURES/TABLES: Labels/captions/legend missing. Illegible text. Overcrowding. Unidentifiable elements (e.g. curves). Poor resolution.	Follows the indicated structure, fair descriptions. No reflections. Document contains some spelling and grammatical errors. Report with more than one page over-length.
2	IF IN AT LEAST ONE FIGURE/TABLE: Labels/captions/legend missing. Illegible text. Overcrowding. Unidentifiable elements (e.g. curves). Poor resolution.	Follows the indicated structure, proper illustration of concepts and results. Poor reflections on results. The document contains little spelling and grammatical errors. Report with one page over-length.
3	Good quality figures/tables. However, information is not properly processed, e.g. several plots could be combined into a single figure for better illustration of main differences. Tables with unnecessary info.	Follows the indicated structure, proper illustration of concepts and results. Limited reflections on results. Document contains only minor spelling and grammatical errors. Report with one page over-length.
4	Good quality figures/tables. However, information is not properly processed, e.g. several plots could be combined into a single figure for better illustration of main differences. Tables with essential info.	Very good in terms of structure, contents, clarity, analysis and reflections. Document is without any spelling and grammatical errors. Report with correct length.
5	Good quality figures/tables. Excellent processing and presentation of results. Only essential information is provided to properly convey the main findings.	Excellent in terms of structure, contents, clarity, analysis and reflections. Useful input for other students. Spelling and grammatically error free. Report with correct length.

Table A.3: Assignment Assessment Matrix

What modeling domains do Smart Grids typically connect?	- Physics, IT, roles, stochastics. - Electricity, Heat, SCADA, Markets. - Energy grids, transport, built environment, law.
What happens at synchronization points (aka macro steps) in a co-simulation?	- Clocks are equalized. - Variables are exchanged. - Output data is collected and aligned.
What can happen in a highly stressed power system?	- Newton Raphson method converges very fast. - Less differential equations needed to model dynamics. - One form of instability may lead to another one.
A generator, that was delivering 200 MW, experiences a sudden frequency rise from 50 Hz to 62 Hz. If the generator does not have any control, what occurs with its kinetic energy?	- Remains unchanged - Decreases by 48 [MJ] - Increases by 107 [MJ] - Decreases by 107 [MJ] - Increases by 48 [MJ]
How does AVR affect the system performance?	- Reduces voltage changes in the system - Reduces frequency changes in the system - Brings the voltage back to pre-disturbance levels
What is the job of the ISO OSI layer 2?	- Drive the voltage on the communication cable - Establish a connection between partners - Arbitrate the communication medium
What is a synchrophasor?	- Synchrophasor is a rotating vector with magnitude and phase with respect to UTC time reference - Synchrophasor is used to time synchronize PMUs - Synchrophasor is a waveform signal, which can be directly measured using a PMU device
Which one is the type of attack that corrupts the integrity of the data?	- False Data Injection (FDI) attack - Distributed denial-of-service (DDoS) attack - Stealthy zero-dynamics attack

Table A.5: Potential exam questions