Robust Optimal Power Flow

Convergence Problems and Solutions

by

Jules Fleuren

Supervisor TU Delft:
Prof.dr.ir. C. Vuik

Supervisor TenneT:
S. Chipli

Thesis Committee Member TU Delft:
Dr. D. de Laat

Thesis Committee Member TenneT:
B. Cijsouw

Project Duration:
October, 2024 - July, 2025

Faculty:
Electrical Engineering, Mathematics and Computer Science, TU Delft

Summary

In the operation of an electricity grid, such as the Dutch transmission system operated by TenneT, decisions need to be made on how to operate components in the grid. These decisions require a complex weighing of the associated costs and effects on the operating state of the grid. Optimal Power Flow (OPF) is a class of mathematical optimization problems that capture the effects of each decision, and aims to find an optimal way of controlling the electricity grid.

OPF is well described in scientific literature, and both open-source and proprietary software exists to perform OPF calculations. However, most implementations are limited in their capabilities. Furthermore, applying existing methods to models of real-world grids is not always straightforward. During the calculations, problems can occur that can either slow down or completely prevent the methods from obtaining a solution. We call these *convergence problems*.

In this thesis, two such types of convergence problems have been identified. The first convergence problem involves the modelling of switches, and the second involves the modelling of parallel transformers. Both problems have been investigated and strategies have been found to resolve these problems.

The result is an OPF method that convergences on a model of the Dutch transmission network. The implementation of this model is very flexible, allowing it to be applied to a wide variety of OPF problems. This is an important step towards the application of OPF in operation of the Dutch transmission grid.

The method has been tested on two different models, one of which is a model of the Dutch transmission grid. Both the computational performance and the resulting solutions have been compared for various OPF calculations.

Preface

This thesis is written to obtain the degree of Master of Science in Applied Mathematics at the Delft University of Technology. It is to be defended on Friday, July 4th 2025 at 15:00.

The energy transition is one of the great challenges of our time. I was therefore very motivated to start this thesis project, even though I did not have much prior knowledge on power systems. I immediately felt right at home with this subject. A topic with both interesting mathematics, and a very direct and relevant application. It was a wonderful experience!

I would like to thank my supervisors, Kees and Shravan for their mentoring and feedback, and for supporting me throughout the project. The rest of the ODINA team: Bastiaan, Joost, Jasper and Parinaz, have also been of great support. I felt right at home with you, and thank you for the encouragement and the time you took to answer my questions. I would additionally like to thank David for taking part in my thesis committee.

Finally, I would like to thank Britte, my parents, and everyone else who supported me during the duration of this project. This thesis is the final chapter of seven wonderful years of studying in Delft, and I am truly grateful for all the people I have met and the experiences I have had during these years.

Jules Fleuren
Delft, June 2025

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
AC	Alternating Current
API	Application Programming Interface
DC	Direct Current
DSO	Distribution System Operator
ILP	Integer Linear Program
LP	Linear Program
MINLP	Mixed Integer Nonlinear Program
NDC	Non Dominated Contingency
NLP	Nonlinear Program
KCL	Kirchoff's Current Law
OPF	Optimal Power Flow
pu	per unit
RMS	Root Mean Square
SCOPF	Security Constrained Optimal Power Flow
TSO	Transmission System Operator

Symbols

Symbol	Definition	Unit
C	number of contingencies	-
$\mathcal{C}$	set of all contingencies	-
$\mathcal{E}$	set of all edges	-
f	objective function	-
$\mathcal{G}$	set of all generators	-
$\mathbf{g}$	vector of equality constraint functions	-
$\mathbf{h}$	vector of inequality constraint functions	-
i	instantaneous current	kA
I	current phasor	kA or pu
$\|I\|$	current magnitude	kA or pu
$I_{\max}$	current amplitude	kA
$I^{\mathrm{f}}_{k,l}$	current flowing out of node k into edge $\{k,l\}$	kA or pu
$I^{\mathrm{t}}_{k,l}$	current flowing into node l out of edge $\{k,l\}$	kA or pu
$I^{\max}_{k,l}$	Upper bound on the current flowing through edge edge $\{k,l\}$	kA or pu
j	imaginary unit	-
L	Lagrange dual function	-
$\mathcal{L}$	set of all loads	-
N	number of nodes	-
$\mathcal{N}$	set of all nodes	-
P	active power	MW or pu
Q	reactive power	Mvar or pu
S	complex power	MVA or pu
t	time	s

Symbol	Definition	Unit
t_{kl}	transformer tap ratio of edge $\{k, l\}$ at side of node k	-
T	system period	s
$\mathbf{u}$	vector of decision variables	-
v	instantaneous voltage	kV
$\mathbf{v}$	vector of state variables	-
V	voltage phasor	kV or pu
V_i	voltage phasor at node i	kV or pu
$\|V\|$	normalized voltage magnitude	kV or pu
$V_{\max}$	voltage amplitude	kV
$\mathbf{w}$	vector of exogenous variables	-
$\mathbf{x}$	vector of optimization variables	-
Y	admittance matrix	Ω^{-1} or pu
Y_{kl}^{es}	shunt admittance of edge $\{k, l\}$	Ω^{-1} or pu
Y_i^{ns}	shunt admittance at node i	Ω^{-1} or pu
Y_{kl}^{sr}	series admittance of edge $\{k, l\}$	Ω^{-1} or pu
Z	impedance	Ω
δ_i	voltage phase shift at node i	rad
θ_{kl}	transformer tap ratio phase shift	rad
τ_{kl}	transformer tap ratio magnitude	-
ϕ	voltage and current phase difference	rad
ψ_I	current phase shift	rad
ψ_V	voltage phase shift	rad
ω	System frequency	Hz
$\nabla_{\mathbf{x}} f$	Gradient of f w.r.t. $\mathbf{x}$	-
$\nabla_{\mathbf{x}}^2 f$ Hessian of f w.r.t. $\mathbf{x}$	-	

1 Introduction

TenneT is a Transmission System Operator (TSO), responsible for the transmission grid in the Netherlands and a large part of Germany. This responsibility includes maintaining, operating, and where necessary, expanding the transmission grid. With the energy transition from fossil fuels to renewable energy sources, which are mostly electric, the demand on the electricity grid increases. In addition to this, a transition from centrally generated energy, with a few big power plants, to distributed generation, with smaller scale solar and wind generation in many places, is taking place. Both of these transitions come with big challenges in the operation and expansion of the grid. To make efficient use of the existing assets and to ensure a stable and reliable operation of the grid, it is crucial to analyse the capabilities of the grid and optimize the control of the grid.

In this chapter, we sketch the position and role of TenneT in the electricity system and give some examples of challenges that TenneT faces in fulfilling these roles.

1.1. Position of TenneT

TenneT is the only TSO in the Netherlands. It is also one of the 4 TSO's operating the German transmission grid, but our focus will be on the Dutch grid. In figure 1.1 a map of the Dutch transmission grid can be seen. As a TSO, TenneT is responsible for the maintenance and operation of the high voltage transmission grid, that is, all parts of the grid that are 110 kV or higher.

Energy is traded on the electricity markets by suppliers and consumers of electricity, and this energy is then transported over the grid. TenneT does not participate in this market by buying or selling energy, but its responsibility is to transport this energy from suppliers to consumers. It can only intervene in the market if the stability of the grid cannot be ensured. These responsibilities and possibilities are determined by law, and TenneT is under supervision of the governmental agency *Autoriteit Consument en Markt*.

Parties that are directly connected to the high voltage transmission grid of TenneT are large power plants, or wind or solar parks with a generation of about 10 MW or higher, large industry with an energy consumption of about 10 MW or higher, and *Distribution System Operators* (DSO) that further distribute electricity to smaller scale industry and homes [41, sec. 2.1].

1.1.1. Ancillary Services

To ensure a stable and reliable transportation of energy, TenneT buys, builds and maintains its own assets, such as transmission lines, substations, and a variety of electrical components. In addition to this, it can however also make use of so-called *Ancillary Services*. These are services provided by parties connected to the grid, that TenneT uses to operate the grid in a stable way. Examples of ancillary services are: providing reserve generation capacity for active power balance and compensation of grid losses, reactive power generation and scaling power generation up or down in case of redispatch. In section 1.2, we elaborate on two situations in which the stability of the grid is at risk and how TenneT makes use of ancillary services in these situations, to ensure the stability of the grid.

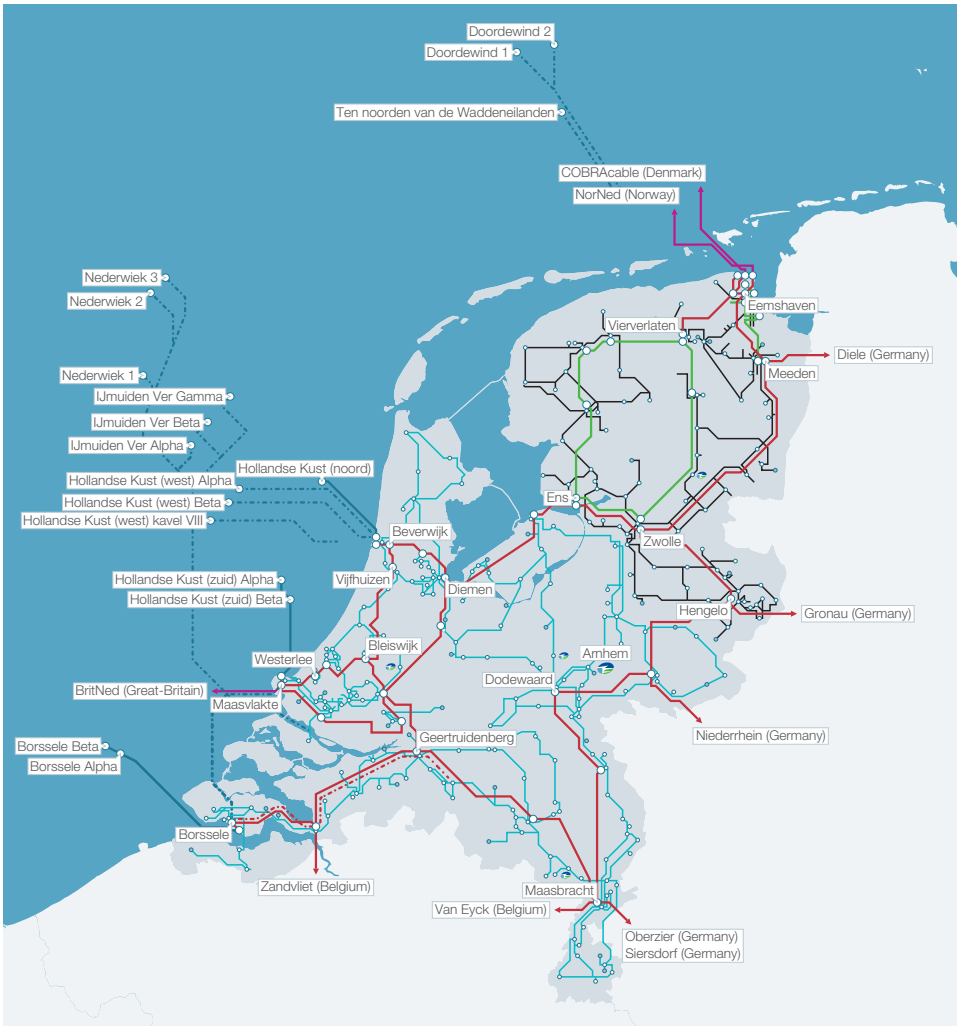

Figure 1.1: Map of the Dutch transmission grid, including connections to neighbouring countries and offshore wind farms. The dashed lines are planned connections. Reproduced from [5].

1.2. Operational Challenges

We introduce two challenges in operating the grid, that TenneT faces on a daily basis. Both challenges illustrate how TenneT has to actively and optimally control the grid, to ensure a stable and reliable transmission grid, at a low cost.

1.2.1. Grid Congestion

When the volume of power transported over the grid is high, and the locations of high supply (generation) and demand (loads) are not well distributed over the grid, *grid congestion* can occur. This means that the grid cannot transport all power without overloading components in the grid. An example of a country where this happens frequently, is Germany, where high amounts of power are generated by offshore wind parks in the north, and a lot of power is consumed by heavy industry in the south. When the grid cannot safely transport all power, a *redispatch* is performed. This means that generation is scaled down in parts of the grid far away from big loads, and power generation located closer to the loads is scaled up. This ensures that all power demand is still satisfied, while relieving some strain on the grid, because the power needs to travel less far. Both the party scaling down their generation and the party scaling up their generation are financially compensated for this. These costs are paid by the TSO, which in turn raises its tariffs to cover the costs. This means that optimally utilizing the grid, and reducing the dependence on redispatch, can reduce costs for all electricity users. In practice, during redispatch, often fossil fuel powered electricity generation is scaled up, and renewable generation is scaled down. This

means that reducing the dependence on redispatch could also reduce strain on the environment, by decreasing greenhouse gas emissions.

1.2.2. Reactive Power Balance

In contrast to active power, reactive power[1] is not traded on the energy market. Making sure supply and demand of reactive power are matched, is a task of TenneT. A lack of supply of reactive power can lead to voltage drops, and a surplus can lead to voltage rises. Voltages that are either too high or too low are undesirable. The task of maintaining this reactive power balance is called *reactive power dispatch*, or *voltage control*. Capacitive and reactive shunts can be activated to either supply or consume reactive power, or ancillary service providers can be asked to supply or consume reactive power. Some ancillary service providers can be asked to maintain a fixed voltage, and thus consume or supply reactive power as needed to maintain this voltage, others can be asked to supply or consume a specified amount of reactive power. Shunts can be activated without additional costs because they are owned by the TSO, but ancillary service providers are compensated financially for the amount of reactive power they provide or consume.

Operating lines and transformers at higher voltages reduces the current required to transmit the same amount of power. Therefore, less power is being lost due to the generation of heat in the components. The power that is lost, is paid for by the TSO. This is an additional factor that should be considered in the operation of the grid.

Determining how shunts should be controlled, and how much reactive power should be requested from ancillary service providers, ensuring stable voltage levels at minimal cost, is a challenging problem.

1.3. Optimal Power Flow

The two examples in Section 1.2 show some of the decisions that TenneT has to make in order to safely and efficiently operate the high voltage grid. Making these decisions is not easy, and a careful consideration of all the available options needs to be made every time. Optimal Power Flow captures this complex problem in a mathematical model, in which the pros and cons of all available options can be weighed against each other. A solution is then calculated, which specifies the choices that need to be made in order to operate the grid in an optimal way, where optimal is according to some specified objective.

[1] For an introduction to active and reactive power, see sec. 3.1.2.

2

Research Goals

In this chapter, we start by sketching the context in which the research is taking place. Then we formulate a research question, and elaborate on the requirements.

2.1. Research Context

At TenneT, there is a need for tools that can do fast, robust, and accurate network analysis calculations, such as power flow calculations, analysis of the grid in contingency situations, and optimal power flow calculations. These calculations sometimes need to be done as individual calculations, and sometimes in bulk, calculating for hundreds of timestamps or different scenarios, or as part of automated processes. For some types of calculations, proprietary software, such as `PowerFactory` (see Section 6.2) satisfies all needs. However, `PowerFactory` is not suited for all uses cases, as is illustrated by the examples in section 6.2.1.

In this context, the `ODINA`[1] toolbox is being developed in-house at TenneT. This is a toolbox for transmission network analysis. The aim of the toolbox is to implement fast and robust network calculations, that can be used in automated processes. The toolbox can interface with `PowerFactory`, so that existing network models that are created in maintained in `PowerFactory` can be imported into the toolbox for calculations. So far, the toolbox is capable of doing fast power flow calculations, with algorithms for bulk DC and AC power flow calculations. The `ODINA` toolbox provides an environment that is well suited for the implementation of an OPF algorithm, since it provides methods to import grid models, calculate admittance matrices, and perform power flow calculations.

2.2. Research Question

Our research question is:

> Can we implement an OPF algorithm, with a focus on speed, robustness, and configurability, that performs well on the Dutch transmission grid?

We elaborate on the requirements on speed, robustness, and configurability.

2.2.1. Requirements on Speed

In operation of the transmission grid, Optimal Power Flow could be used for calculating optimal set-points for controllable components. In order for the result of OPF calculations to be used in real-time operation of the grid, calculations have to be done in a limited timespan. Predictions of power generation and consumption are not always known very far in advance, and reconfiguration of the grid is done on an hourly basis. Therefore, calculations on the full grid, for 24 hourly timestamps, should ideally not take longer than an hour.

In planning of the grid, OPF could be used to analyse possible configurations of the grid, and calculate

[1] Open Developer Initiative for Network Analysis

associated operating costs for many operation scenarios. To be able to effectively do these kinds of analyses, TenneT should be able to perform OPF calculattions in bulk.

2.2.2. Requirements on Robustness

An important condition for the implementation of OPF calculations in business processes of TenneT, is that the algorithms should be robust. The methods should work on a broad range of grid variations and operating scenarios. Most importantly, it should always find feasible solutions, and although (proven) strict optimality might not be realistic, results should always be near optimal. In the case that it cannot be prevented that the method fails in some cases, it should clearly indicate to the user what went wrong and suggest solutions. These requirements on robustness are focussed on models of the Dutch grid. Robust convergence on grid models that are not similar to the Dutch grid would be an added benefit, but this is of secondary importance.

2.2.3. Requirements on Configurability

As we will describe in chapter 5, Optimal Power Flow is a broad category of problems, and it can vary between applications what the decision variables and constraints are and what the objective is. The aim is to make an OPF module that is easy to configure for a broad range of types of OPF problems, so that it is widely applicable. Furthermore, it should support bulk calculations, where calculations are done with different power injection scenarios, grid topology scenarios, and contingency scenarios. The support of configuration options may be restricted to use cases that are specific to the Dutch grid.

3

Modelling the Transmission Grid

To be able to mathematically analyse the grid, and to predict its behaviour, we introduce a mathematical model of a transmission system. First we introduce some concepts from the theory of Alternating Current (AC) circuits, then we describe how we can model a transmission grid as graph, and finally, we introduce *contingencies*, an important concept in the operation and modelling of transmission grids.

3.1. AC Circuit Fundamentals

Since the transmission of power via the power grid is mostly done with alternating current, we give an introduction to the behaviour of AC-circuits and how they can be modelled. We mostly follow the reasoning and notation of [33], [45] and [18].

3.1.1. Voltage and Current

In an AC-circuit, voltage and current vary in time following a sinusoidal function. In steady state circuit analysis, we assume that the frequency of these functions is fixed. Consider any part of a transmission network, that is connected to the rest of the network at two terminals, such as a cable or a tranformer. We can consider the voltage across the two terminals and the current through the component as a function of time. The voltage is as follows:

$$v(t) = V_{\max} \cos(\omega t + \psi_V) \tag{3.1}$$

and the current is as follows:

$$i(t) = I_{\max} \cos(\omega t + \psi_I). \tag{3.2}$$

Here,

$$\begin{aligned} V_{\max} &= \text{voltage amplitude, kV} \\ I_{\max} &= \text{current amplitude, kA} \\ \omega &= \text{angular frequency, Hz} \\ t &= \text{time, s} \\ \psi_V &= \text{voltage phase shift, rad} \\ \psi_I &= \text{current phase shift, rad.} \end{aligned}$$

The time is measured to some reference time, where $t = 0$. In Europe, the electricity grid operates at a frequency of 50 Hz, this means that the angular frequency is $\omega = 2\pi \cdot 50\,\text{Hz}$. We define $\phi = \psi_V - \psi_I$ as the phase difference between the voltage and current. If ϕ is positive, we say that the voltage is leading the current and if ϕ is negative, the voltage is lagging the current.

In equations (3.1) and (3.2), $I_{\max}$ and $V_{\max}$ denote the amplitude of the current and voltage, respectively. However, in the context of AC electronics, it is customary to work with the Root Mean Square (RMS) of

power and current [45, p. 7]. They can be calculated as follows:

$$|V| = \sqrt{\frac{1}{T}\int_0^T v(t)^2 dt} \tag{3.3}$$

$$|I| = \sqrt{\frac{1}{T}\int_0^T i(t)^2 dt} \tag{3.4}$$

here $T = 2\pi/\omega$ is the period of the sine waves. Substituting (3.1) and (3.2) in these formulas gives the relations $V_{max} = \sqrt{2}|V|$ and $I_{max} = \sqrt{2}|I|$.

Phasor Notation
If we rewrite equations (3.1) and (3.2) using Euler's identity, we get the following:

$$\begin{aligned} v(t) &= \sqrt{2}|V|\cos(\omega t + \psi_V) \\ &= \sqrt{2}\,\mathrm{Re}\left\{|V|e^{j\psi_V}e^{j\omega t}\right\} \\ &= \sqrt{2}\,\mathrm{Re}\left\{Ve^{j\omega t}\right\} \end{aligned} \qquad \begin{aligned} i(t) &= \sqrt{2}|I|\cos(\omega t + \psi_I) \\ &= \sqrt{2}\,\mathrm{Re}\left\{|I|e^{j\psi_I}e^{j\omega t}\right\} \\ &= \sqrt{2}\,\mathrm{Re}\left\{Ie^{j\omega t}\right\}. \end{aligned}$$

Here $V = |V|e^{\psi_V}$ and $I = |I|e^{\psi_I}$. Note that these equations can be used both ways, so we obtain the following one-to-one relationship between the complex numbers, and steady state voltage functions:

$$V = |V|e^{j\psi_V} \qquad \leftrightarrow \qquad v(t) = \sqrt{2}|V|\cos(\omega t + \psi_V) \tag{3.5}$$

and, similarly, between the complex numbers and steady state current functions:

$$I = |I|e^{j\psi_I} \qquad \leftrightarrow \qquad i(t) = \sqrt{2}|I|\cos(\omega t + \psi_I). \tag{3.6}$$

We call V and I the *voltage* and *current phasors*, respectively. In the rest of this report, we will denote the steady state voltage and current functions at some point in the grid by its voltage and current phasors. For more information, and a derivation of calculation rules for phasors, we refer to [45, sec. 1.4].

3.1.2. Power

The instantaneous power $p(t)$ consumed by a load through which an alternating current $i(t)$ flows across a voltage of $v(t)$, is calculated as follows:

$$p(t) = v(t)i(t) \tag{3.7}$$

$$= \sqrt{2}|V|\cos(\omega t + \psi_V)\sqrt{2}|I|\cos(\omega t + \psi_V - \phi) \tag{3.8}$$

$$= |V||I|\cos(\phi)(1 + \cos(2(\omega t + \psi_V))) + |V||I|\sin(\phi)\sin(2(\omega t + \psi_V)) \tag{3.9}$$

$$= P(1 + \cos(2(\omega t + \psi_V))) + Q\sin(2(\omega t + \psi_V)) \tag{3.10}$$

here $P = |V||I|\cos(\phi)$ and $Q = |V||I|\sin(\phi)$. We call P *active power* or *real power*, and we call Q *reactive power* or *imaginary power*.

We see that active power and reactive power are dependent on the phase difference ϕ between voltage and current. Furthermore, both terms oscillate with a frequency of $2\omega t$, but the first term $P(1+\cos(2\omega t))$ is unidirectional with an average value of P and the second term $Q(\sin(2\omega t))$ is bidirectional with an average of 0. If $\phi > 0$, then $Q > 0$ and we say that the load consumes reactive power, conversely, if $\phi < 0$, then $Q < 0$ and we say the load supplies reactive power.

We define the *complex power* $S = P + jQ$, and we can calculate this using the voltage and current phasors with the following formula:

$$S = VI^* \tag{3.11}$$

where $\cdot^*$ denotes complex conjugation. The apparent power $|S|$ is the magnitude of the current:

$$|S| = |V||I| = \sqrt{P^2 + Q^2}. \tag{3.12}$$

For complex power and apparent power we use the unit MVA, for active power we use the unit MW and for reactive power we use the unit Mvar.

3.1.3. Impedance and Admittance

In an AC circuit, every component has an impedance, which characterizes the opposition that the component poses to current. The impedance is expressed as a complex number $Z = R + jX$ with unit Ω. Here, the real component R is the resistance and the imaginary component X is the reactance. If a component is inductive with an inductance of L Henry ($L > 0$), we have that $X > 0$ and $jX = j\omega L$. If a component is capacitive with a capacitance of C Farad ($C > 0$), we have $X < 0$ and $jX = 1/j\omega C$. The canonical examples of inductive and capacitive loads are the inductor (or coil) and capacitor.

We call the reciprocal of the impedance the *admittance*, and denote it by Y. The real and imaginary components of the admittance are called the *conductance* and *susceptance*. We write $Y = G + jB$. The conductance and susceptance can be calculated with the following formula:

$$G + jB = (R + jX)^{-1} = \frac{Z^*}{ZZ^*} = \frac{R}{R^2 + X^2} - j\frac{X}{R^2 + X^2}. \tag{3.13}$$

When we view impedance as a complex number and combine this with the phasor notation for voltages and currents, we get a useful extension of Ohm's law for steady state AC-circuits [45, sec. 1.4.1], namely:

$$V = ZI \text{ or } I = YV. \tag{3.14}$$

Here, V is the phasor of the voltage across the component and I the phasor of the current flowing through the component.

Say we have a component with impedance $Z = R + jX$ with $R > 0$, and across this component is a voltage V, which is purely real (i.e., $\psi_V = 0$). Then the current looks as follows:

$$I = YV = V\left(\frac{R}{R^2 + X^2} - j\frac{X}{R^2 + X^2}\right), \tag{3.15}$$

or in polar notation:

$$|I| = \frac{|V|}{R^2 + X^2} \text{ and } \phi = -\operatorname{Arg}(I) = \arctan(X/R) \tag{3.16}$$

Hence, a purely resistive load only influences the amount of current flowing, and the current will be in phase with the voltage (i.e., if $X = 0$ then $\phi = 0$). If, however, the reactance is non-zero, then the current will not be in phase with the voltage. An inductive load, with positive reactance, causes the current to lead the voltage and consumes reactive power (i.e., if $X > 0$, then $\phi > 0$). Conversely, a capacitive load causes the current to lag the voltage and supplies reactive power (i.e., if $X > 0$, then $\phi > 0$). Therefore, an inductive load consumes reactive power and a capacitive load supplies reactive power.

3.1.4. Nominal Voltage and Per Unit Normalization

Each component in an electricity grid is assigned a nominal voltage level, which is the voltage that the component is designed to operate at. Typically, the grid consists of a few different nominal voltage levels, and each part of the grid is assigned one of the levels. The physical equipment also needs to be rated for this voltage level. The Dutch grid, for example, has four voltage levels in the transmission grid: 110 kV, 150 kV, 220 kV and 380 kV (line-to-line). In practice, voltages will not be exactly equal to the nominal voltages, but will vary slightly around this nominal voltage.

In the study of power systems, it is common to normalize certain numerical values by dividing them by some fixed base quantity [45, sec. 1.8]. The values are then not given in their normal unit, but "per-unit". This normalized quantity is a dimensionless quantity, and we denote it by "pu" (e.g., $|V| = 1.01$ pu). This is often done with voltages, currents, impedances, and powers. Since these quantities are related via (3.11) and (3.14), choosing a base for two of these quantities determines the other two. We choose a base for voltage and power.

For voltages, we select the nominal (line-to-line) voltage as a base, so that in normal operating conditions, we expect the normalized voltage magnitude $|V|$ to be close to 1 pu. This makes it easy to check if the voltages are within normal operating conditions, at a glance, without needing to compare them to the nominal voltage level. The power exchanged by parties connected to the transmission grid is usually on the order of 100 MVA, so we pick that as the basis for power. This fixes the base for impedances

and currents. Suppose the nominal voltage is $\{V^{\text{nom}}\}$ kV, then converting a current from pu to kA can be done by multiplying with $\sqrt{3}\{V^{\text{nom}}\}$, and converting an impedance from pu to Ω can be done by multiplying with $\{V^{\text{nom}}\}^2$. We refer to [45, sec. 1.8] for some examples.

3.1.5. Three Phase Power System

In AC transmission systems, power is usually provided in a three-phase system. However, in a balanced three-phase system, we may instead consider an equivalent single-phase system [45, sec. 1.7]. These networks can be represented in a *one-line diagram*, which only shows a single line per three-phase connection.

Voltages in a three-phase system can be measured either line-to-neutral, or line-to-line. In power systems, the convention is to use line-to-line voltages.

3.2. The Grid as a Graph

We can model the transmission grid as an undirected graph of points in the electrical grid (nodes) that are connected by electrical components (edges). In this section, we describe how the different components in the grid are represented in this graph model.

All points in the grid that satisfy one of the following characteristics, are nodes: all points that directly connect to more than two lines, points that are directly connected to components such as transformers, loads, generators or shunts, or points where specific quantities (e.g., voltage or current flow), are of interest. In the context of transmission grids, nodes are called *buses*, since they usually represent a physical busbar. We will use the terms bus and node interchangeably. The edges of the graph represent connections between the buses, either in the form of high voltage lines or in the form of transformers. Additionally, at each node, we can define a *power injection*. Components such as generators, loads, and shunts are modelled as components that inject power at a specific bus.

We denote the set of all nodes by $\mathcal{N}$, and the set of all edges by $\mathcal{E}$. We denote the graph by $(\mathcal{N}, \mathcal{E})$. Furthermore, we define $N = |\mathcal{N}|$, the total number of nodes in the grid. When we talk about the *topology* of the grid, we mean the graph $(\mathcal{N}, \mathcal{E})$. For example, we might say: "the topology of scenario A and scenario B is the same", which would mean that the graph in the model of scenario A and scenario B is the same[1].

3.2.1. Nodes

In this section, we describe what buses exactly are, the types of buses there are, and the characteristics of these types.

A bus represents a single component in the system. Physically, this point can be a busbar at a substation, but it can also be a (hardwired) three-way junction somewhere in a transmission tower, a place where a DSO or a customer connects to the grid, or a connection to another grid. Furthermore, a physical substation can also include multiple buses, and the number of buses at a substation can depend on the configuration of switches.

Power Injection and Current Injection

Generators and loads that are directly connected to a bus are modelled as *power injections*. The power injection is the complex power that is supplied or consumed by the load or generator and is made up of the active power injection and reactive power injection. The convention is that, when active (or reactive) power is generated, the active (or reactive) power injection is positive, and when it is consumed, the power injection is negative.

If some load or generator injects power at a node, this is of course in the form of a current at a specific voltage. If the injected power is S and the voltage at the node is V (in phasor form), then the *current injection* is the current flowing in or out of the generator or load. It can be calculated via (3.11).

Although shunts can also be considered as a component that injects power, we treat them separately in section 3.2.3.

[1] In this quote, we do not mean that the admittance matrix (see section 4.1) is the same between in scenario A and scenario B.

Voltage and Power Injection
Each bus has four quantities that are of importance. For bus i, these are: total active power injection P_i, total reactive power injection Q_i, voltage magnitude $|V_i|$ and voltage angle δ_i. If multiple loads and generators are attached to the bus, P_i and Q_i are the net generation or consumption of active and reactive power. Since we assume all loads and generators are attached to the bus in parallel, there can only be one bus voltage. The voltage magnitudes are in pu, and the voltage angle is the voltage phase shift (or the angle of the voltage phasor). We can also treat voltage and power injection in their complex form: $V_i = |V_i|e^{j\delta_i}$ and $S_i = P_i + jQ_i$. If these four quantities are known at every bus, the whole state of the system is known.

If the four quantities $P_i, Q_i, |V_i|$ and δ_i are known at each bus i and the admittance of all components is known, then all currents flowing through edges, and currents flowing in and out of buses can be calculated. Therefore, the currents do not need to be known, to specify the full state of the system.

Types of Buses
Depending on the type of bus, some of these quantities can be directly controlled, or are fixed, and are therefore always known. Others are dependent on the interaction with other parts of the grid. For all buses, two of the four quantities are known, and two are unknown.

PQ Bus Loads do not have any control over voltage levels, but their active power and reactive power consumption are known. These types of buses are therefore called *PQ buses*. Depending on how they are connected to the grid, buses that have photovoltaic cells, wind turbines or certain types of generators[2] attached to them can also behave as PQ buses, so P_i and Q_i are not necessarily negative. A bus with neither generators nor loads attached is modelled as a PQ bus with $P_i = Q_i = 0$.

PV Bus *Synchronous generators*[3] are generators that have control over their output voltage and over their active power output. These buses are therefore called *PV buses*. Any bus that has both loads (i.e. components where the real and reactive power injections are known) and synchronous generators attached to them is also considered as a PV bus.

Slack Bus Due to the conservation of energy, the sum of all power injected plus all power lost (e.g. in transmission lines) should be 0. It is, however, not possible to know the exact amount of losses beforehand. Therefore, we always assign at least one, so-called, *Slack bus* in every grid. This bus is typically a synchronous generator, where we do not fix the power output P_i, but allow it to vary, to make sure this total sum of power is 0. In this way, it "picks up the slack", which explains the name. For a slack bus, the voltage level $|V_i|$ and voltage angle δ_i are fixed. This also defines a reference for voltage angles, based on which all the angles are calculated. In some cases, the slack bus takes the form of a connection to an external grid[4] instead of a synchronous generator.

3.2.2. Edges
Edges are connections between the buses. Physically, these are overhead transmission lines, underground cables, transformers, or (very occasionally) series capacitors or series reactors. All edges have a series impedance, a shunt impedance, and in the case of transformers, also a tap ratio. All edges are modelled via the same model, which is described in appendix A.

Voltage and current over and through a general edge $\{k, l\}$ obey the following equation:

$$\begin{bmatrix} I^{\mathrm{f}}_{kl} \\ -I^{\mathrm{t}}_{kl} \end{bmatrix} = \begin{bmatrix} \frac{1}{|t_{kl}|^2}\left(Y^{\mathrm{sr}}_{kl} + \frac{Y^{\mathrm{es}}_{kl}}{2}\right) & -\frac{Y^{\mathrm{sr}}_{kl}}{t_{kl}{}^{*} t_{lk}} \\ -\frac{Y^{\mathrm{sr}}_{kl}}{t_{kl} t_{lk}{}^{*}} & \frac{1}{|t_{lk}|^2}\left(Y^{\mathrm{sr}}_{kl} + \frac{Y^{\mathrm{es}}_{kl}}{2}\right) \end{bmatrix} \begin{bmatrix} V_k \\ V_l \end{bmatrix}. \tag{3.17}$$

[2] In the rest of this thesis, we use the term *PQ generator* to refer to any power source that has control over both active and reactive power output.

[3] In the rest of this thesis, we use the term *PV generator* to refer to any power source that has control over both active power output and voltage level.

[4] E.g., a connection to the grid of a neighbouring country, or, when only part of a bigger grid is modelled, a connection to the rest of the grid.

Here[5],

$$\begin{aligned} I^{\mathrm{f}}_{kl} &= \text{current flowing out of node } k \\ I^{\mathrm{t}}_{kl} &= \text{current flowing into node } l \\ t_{kl}, t_{lk} &= \text{complex tap transformer ratios} \\ Y^{\mathrm{sr}}_{kl} &= \text{series admittance} \\ Y^{\mathrm{es}}_{kl} &= \text{edge shunt admittance.} \end{aligned}$$

A derivation of this formula can also be found in appendix A. We remark that $I^{\mathrm{f}}_{kl} = -I^{\mathrm{t}}_{lk}$. The complex tap ratio t_{kl} has magnitude τ_{kl}, and angle θ_{kl}. The latter is also referred to as the tap ratio *phase shift*. We have two tap ratios, to model a transformer with a tap on both the high and low voltage side. With t_{kl}, we denote the ratio of the tap on the side of node k, and with t_{lk}, the ratio of the tap on the side of node l. When an edge is a transmission line, $t_{kl} = t_{lk} = 1$. Note that the matrix in (3.17) is only symmetric if t_{kl} and t_{lk} are real.

In a real-life electricity grid, there might be parallel lines or parallel transformers, that connect the same two nodes. However, in our model, we only allow one edge per pair of nodes, as is common for mathematical graphs. When there are parallel lines or transformers, they are combined into a single edge. The matrix in (3.17) for such an edge can then be obtained by summing the admittances of the individual lines together.

Usually, we split I^{f}_{kl} in the following two terms:

$$I^{\mathrm{f}}_{kl} = \left(\frac{V_k}{t_{kl}} - \frac{V_l}{t_{lk}}\right)\frac{Y^{\mathrm{sr}}_{kl}}{{t_{kl}}^*} + V_k\frac{Y^{\mathrm{es}}_{kl}}{2|t_{kl}|^2} \tag{3.18}$$

and indicate the first term by I_{kl}:

$$I_{kl} = \left(\frac{V_k}{t_{kl}} - \frac{V_l}{t_{lk}}\right)\frac{Y^{\mathrm{sr}}_{kl}}{{t_{kl}}^*}. \tag{3.19}$$

The two complex tap ratios t_{kl} and t_{lk} of a transformer, are a function of a real variable, called the *tap position variable*. We denote the tap variable by t^{pos}_{kl}. It would be more accurate to write t_{kl} and t_{lk} as functions of this single tap variable:

$$t_{kl} = t_{kl}(t^{\mathrm{pos}}_{kl}) \tag{3.20}$$

$$t_{lk} = t_{lk}(t^{\mathrm{pos}}_{kl}). \tag{3.21}$$

3.2.3. Shunts

A shunt is a connection between a bus and ground. Shunts are installed in the power grid by TSO's to either supply or consume reactive power, and can often be connected or disconnected when needed. Just like a generator or load, it injects power at a node. However, unlike a generator or load, the amount of power it injects is always dependent on the voltage at the node. This is why we cannot treat them the same as loads or generators. The shunt typically has a conductance that is (almost) zero, and it is either capacitive or inductive. We denote the admittance of a shunt connected to node i with Y^{ns}_i ("ns" short for "node shunt").

Through a shunt flows the following current, denoted by I^{ns}_i:

$$I^{\mathrm{ns}}_i = V_i Y^{\mathrm{ns}}_i. \tag{3.22}$$

3.2.4. Switches and Breakers

In the physical electricity grid, there are switches and breakers that can connect and disconnect all types of components. In [50, sec. III.D] some different ways of modelling switches are described.

[5]Here, "f", "t","sr" and "es" are short for "from", "to", "series" and "edge shunt", respectively.

We highlight a particular choice that can be made when modelling switches that do not connect to a line or a transformer, but rather a switch that connects buses. Physically these are for example busbar couplers. There are two common ways of modelling a grid with such a switch.

First, we can model such a switch as an edge with a very high admittance. This very admittance can either be based on the admittance of the physical switch, or in some cases when this information is not available, it is set to a very large number.

Secondly, it is also possible to represent the two components that the switch connects as a single node. In operation of the grid, we expect the voltages at either end of the switch to be very close together, so modelling them as a single node is a reasonable assumption.

A downside of the first method, is that it introduces edges to the model that have an admittance that is many orders of magnitude bigger than other edges in the grid. This can lead to issues when performing calculations on the grid. The second method does not have this downside, but it has another downside. In some situations, especially for models that are used in operational scenarios, it might be relevant to know how much current is flowing through a switch. With the first method, the current is calculated as part of a power flow calculation (see chapter 4), with the second method however, this information is lost.

3.3. Contingencies

During the operation of the transmission grid, sometimes equipment fails. We call this a *contingency*. To assure a stable and uninterrupted operation of the grid, even in the case of contingencies, it is necessary to study how the grid would behave if these contingencies were to occur. Common contingencies that are studied are: the outage of a line or transformer, the outage of a shunt and the outage of a generator.

In this report, we assume there is a pre-defined list of contingencies that might occur. Say we have a list of C contingencies, then we index them by the integers $1, \ldots, C$, and we denote the set of all contingencies by $\mathcal{C}$. The base situation, where no contingency has occurred, has index 0. The base situation is also called the *N-0 state*, and when a single contingency has occurred, we are in an *N-1 state*[6]. Here the "N" stands for normal or nominal, and the "-1" means that there is a single contingency.

3.3.1. N-1 Secure

When the grid is operating in its N-0 state, without any unplanned outages, TSO's usually operate their grid in such a way that it is *N-1 secure*. This means that if any single piece of equipment fails, the grid should still be able to operate normally, within all safety margins. For planning and design of the grid, this often means that the grid is designed to be N-2 secure, so that equipment outages can be planned (e.g., for maintenance) all while the grid is still operating in an N-1 secure state.

[6] We can of course extend this to the general *N-k state* for any integer k.

4

Power Flow Modelling

In chapter 3, we introduced equations that describe the behaviour of individual components, and we described how we can model a transmission network as a graph. In this chapter, we first introduce the admittance matrix, then we derive a system of equations that describes the interaction between all components in the grid. In section 4.3.2, we describe how to solve this system, to obtain an operating state of the grid.

4.1. Admittance Matrix

We now construct the so-called *Admittance Matrix*, Y. This is a complex $N \times N$ matrix, where component Y_{kl} is defined as follows:

$$Y_{kl} = \begin{cases} -\frac{Y_{kl}^{\text{sr}}}{t_{kl}{}^{*} t_{lk}} & \text{if } \{k,l\} \in \mathcal{E} \\ Y_k^{\text{ns}} + \sum\limits_{\substack{i \in \mathcal{N} \\ i \neq k}} \frac{1}{|t_{ki}|^2} \left(Y_{ki}^{\text{sr}} + \frac{Y_{ki}^{\text{es}}}{2} \right) & \text{if } k = l \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

If we compare this definition with equation (3.17), we see some similarities. Where the matrix in equation (3.17) describes the voltage and current for a single edge, this matrix combines all the interactions by summing them together[1]. Furthermore, we add the shunt admittance to the diagonal elements.

The admittance matrix is, in general, not symmetric, since Y_{kl} can differ from Y_{lk} if t_{kl} is not real. For large grids, the matrix is usually quite sparse, since the number of elements in row or column i is the number of edges that connect to i, plus 1. Note that for an edge $\{k,l\}$ in $\mathcal{E}$, equation (3.19) works out to:

$$I_{kl} = Y_{kl}(V_l - V_k). \tag{4.2}$$

This equation actually holds for all distinct pairs of nodes, whether there is an edge connecting them or not, because if $\{l,k\} \notin \mathcal{E}$, then $I_{lk} = 0$ and $Y_{kl} = 0$, so it also holds.

4.2. Power Flow Equations

In this section we derive the *power flow equations*, a system of equations relating the four quantities $|V_i|$, δ_i, P_i and Q_i at all nodes $i \in \mathcal{N}$. We mostly follow the reasoning of [18].

[1] To make this precise, we can say that the matrix in (3.17) contains just the four components with indices $(k,k), (k,l), (l,k), (l,l)$, of a larger $N \times N$ matrix, with all other components 0. These larger matrices are then summed together.

4.2.1. Current at a Single Node

We now consider a node k with current injection I_k. By *Kirchhoff's Current Law* (KCL), the injected current is equal to the sum of all current flowing out of the node, either via an edge or via a shunt:

$$I_k = I_k^{\text{ns}} + \sum_{\substack{l \in \mathcal{N} \\ l \neq k}} I_{kl}^{\text{f}}. \tag{4.3}$$

Here I_{kl}^{f} is the current flowing out of node k into edge k, l, if $\{k, l\}$ is an edge (see (3.17)) and $I_{kl}^{\text{f}} = 0$ if $\{k, l\}$ is not an edge. Now we use (3.17) and (3.22):

$$\begin{aligned}
I_k &= I_k^{\text{ns}} + \sum_{\substack{l \in \mathcal{N} \\ l \neq k}} I_{kl}^{\text{f}} && (4.4)\\
&= Y_k^{\text{ns}} V_k + \sum_{\substack{l \in \mathcal{N} \\ l \neq k}} \left[\frac{1}{|t_{kl}|^2} \left(Y_{kl}^{\text{sr}} + \frac{Y_{kl}^{\text{es}}}{2} \right) V_k - \frac{Y_{kl}^{\text{sr}}}{t_{kl}{}^* t_{lk}} V_l \right] && (4.5)\\
&= \left(Y_k^{\text{ns}} + \sum_{\substack{i \in \mathcal{N} \\ i \neq k}} \frac{1}{|t_{ki}|^2} \left(Y_{ki}^{\text{sr}} + \frac{Y_{ki}^{\text{es}}}{2} \right) \right) V_k + \sum_{\substack{l \in \mathcal{N} \\ l \neq k}} -\frac{Y_{kl}^{\text{sr}}}{t_{kl}{}^* t_{lk}} V_l && (4.6)\\
&= Y_{kk} V_k + \sum_{\substack{l \in \mathcal{N} \\ l \neq k}} Y_{kl} V_l && (4.7)\\
&= \sum_{l \in \mathcal{N}} Y_{kl} V_l && (4.8)
\end{aligned}$$

In the third step we used (4.1). If we define the vectors $\mathbf{I}^{\text{inj}} = (I_1, \ldots, I_N)$ and $\mathbf{V} = (V_1, \ldots, V_N)$, then in matrix notation we can rewrite this to $\mathbf{I}^{\text{inj}} = Y\mathbf{V}$.

4.2.2. Power Injection

Equation (4.8) relates the current injection at each node to the nodal voltages, via the admittance matrix. However, we would like to relate the power injection at each node to the nodal voltages. To do this we take the conjugate of (4.8) and multiply both sides by V_k:

$$\begin{aligned}
S_k &= V_k I_k{}^* && (4.9)\\
&= V_k \left(\sum_{l \in \mathcal{N}} Y_{kl} V_l \right)^* && (4.10)\\
&= \sum_{l \in \mathcal{N}} V_k V_l{}^* Y_{kl}{}^* && (4.11)\\
&= \sum_{l \in \mathcal{N}} |V_k||V_l| e^{j(\delta_k - \delta_l)} \left(G_{kl} - jB_{kl} \right) && (4.12)\\
&= \sum_{l \in \mathcal{N}} |V_k||V_l| (\cos(\delta_k - \delta_l) + j \sin(\delta_k - \delta_l)) \left(G_{kl} - jB_{kl} \right). && (4.13)
\end{aligned}$$

Here we split Y_{lk} into its real and imaginary components $Y_{ik} = G_{ik} + jB_{ik}$. Considering the real and complex components separately gives us:

$$\begin{aligned}
P_k &= \sum_{l \in \mathcal{N}} |V_k||V_l| (G_{kl} \cos(\delta_k - \delta_l) + B_{kl} \sin(\delta_k - \delta_l)) && (4.14)\\
Q_k &= \sum_{l \in \mathcal{N}} |V_k||V_l| (G_{kl} \sin(\delta_k - \delta_l) - B_{kl} \cos(\delta_k - \delta_l)). && (4.15)
\end{aligned}$$

These equations are commonly known as the *Power Flow Equations*.

4.3. Power Flow Problem

We now state the *Power Flow problem* (sometimes also known as the *Load Flow problem*). For a network with admittance matrix Y, given at every node i two out of the four variables $|V_i|, \delta_i, P_i, Q_i$, find the remaining variables, such that equations (4.14) and (4.15) are satisfied for all k in N. The variables that are known and unknown can be seen in table 4.1.

Bus type	Known variables	Unknown variables
PQ bus	P_i, Q_i	$\|V_i\|, \delta_i$
PV bus	$\|V_i\|, P_i$	δ_i, Q_i
Slack bus	$\|V_i\|, \delta_i$	P_i, Q_i

Table 4.1: Known and unknown variables for node i in the power flow problem, depending on the bus type.

This problem is a nonlinear system of $2N$ equations with $2N$ known, and $2N$ unknown variables. A solution of the Power Flow Problem is sometimes called a *power flow* or a *load flow*.

As we mentioned in section 3.2.1, once the four quantities $|V_i|, \delta_i, P_i$ and Q_i are known at each bus i, we know the whole state of the system, and the currents through each edge can be calculated from these variables.

In the rest of this thesis, we assume that the stated power flow problems have a unique solution. For more information on this assumption, we refer to [18, ch. 7].

4.3.1. Equivalent Formulations

Note that we have stated the problem here in terms of the voltage magnitude and angle (polar coordinates), and the real and imaginary component of the power injection (Cartesian coordinates). We can, however, also state the problem in terms of complex voltage and power, and complex power injection, or any combination of either Cartesian or polar coordinates of both variables. Furthermore, we could also state the problem in terms of complex voltage and complex current, satisfying (4.8). The most widely used formulation is the formulation as stated here, but for a comparison of the formulations, and their advantages and disadvantages in solving the power flow problem, we refer to [46].

4.3.2. Power Flow Solvers

Finding the state of an electricity grid, equates to solving a nonlinear system of $2N$ equations for $2N$ unknowns. There are multiple numerical algorithms for solving this problem, both for obtaining accurate and approximate solutions. Some of these methods are: the Newton-Raphson method, the Fast Decoupled Load Flow method and DC approximation method [18, ch. 4]. We describe the most common method, via the Newton-Raphson algorithm, in section 4.3.3. We will not treat the DC approximation method here, but there are Optimal Power Flow methods (see chapter 5) that build upon this algorithm, so for a treatment of this method, we refer to [18, sec. 4.3].

4.3.3. Newton Raphson Method

In this section we describe the Newton-Raphson method for solving the power flow problem. We mostly follow the reasoning of [18].

We can solve the system with the Newton-Raphson method. Let $\boldsymbol{\delta}$ be a vector containing all unknown voltage angles (i.e., δ_i for all PV- or PQ buses) and let $\mathbf{|V|}$ be a vector containing all unknown voltage magnitudes (i.e., $|V_i|$ for all PQ buses). Let $\mathbf{x}$ be a vector containing both:

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\delta} \\ \mathbf{|V|} \end{bmatrix}. \tag{4.16}$$

Now we define the active and reactive power-mismatch functions ΔP_i and ΔQ_i as follows:

$$\begin{aligned}\Delta P_i(\mathbf{x}) &= P_i^{\text{sp}} - P_i && (4.17)\\ &= P_i^{\text{sp}} - \sum_{l\in\mathcal{N}} |V_i||V_l| \left(G_{il}\cos(\delta_i - \delta_l) + B_{il}\sin(\delta_i - \delta_l)\right) && (4.18)\\ \Delta Q_i(\mathbf{x}) &= P_i^{\text{sp}} - P_i && (4.19)\\ &= Q_i^{\text{sp}} - \sum_{l\in\mathcal{N}} |V_i||V_l| \left(G_{il}\sin(\delta_i - \delta_l) - B_{il}\cos(\delta_i - \delta_l)\right). && (4.20)\end{aligned}$$

Here, P_i^{sp} and P_i^{sp} are the specified active and reactive power injection, and P_i and Q_i are the computed power injection that follow from (4.14) and (4.15). Note that ΔP_i can only be computed if i is a PV- or PQ bus, and ΔQ_i only if i is a PQ bus, because otherwise, P_i^{sp} or Q_i^{sp} is not given. Now we construct the total power mismatch function as follows:

$$\mathbf{F}(\mathbf{x}) = \begin{bmatrix}\Delta\mathbf{P}(\mathbf{x})\\ \Delta\mathbf{Q}(\mathbf{x})\end{bmatrix}, \tag{4.21}$$

where $\Delta\mathbf{P}$ and $\Delta\mathbf{Q}$ are the vectors containing all ΔP_i and ΔQ_i that can be computed. We remark that $\mathbf{F}(\mathbf{x}) = 0$, if and only if, $\mathbf{x}$ solves the system for all unknown voltage magnitudes and angles. Furthermore, the remaining unknowns (all of the form P_i or Q_i) can be found by using (4.14) and (4.15). Therefore, we have a solution of the power flow problem, if and only if, we have found $\mathbf{x}$ such that $\mathbf{F}(\mathbf{x}) = 0$. Solving the Power Flow Problem can therefore be done by finding a zero of $\mathbf{F}$.

Finding this zero of $\mathbf{F}$ can be done using the Newton-Raphson algorithm. Each Newton-Raphson iteration then requires solving the following system to obtain the iteration step:

$$-\begin{bmatrix}\frac{\partial\Delta\mathbf{P}}{\partial\delta} & \frac{\partial\Delta\mathbf{P}}{\partial|V|}\\ \frac{\partial\Delta\mathbf{Q}}{\partial\delta} & \frac{\partial\Delta\mathbf{Q}}{\partial|V|}\end{bmatrix}\begin{bmatrix}\Delta\boldsymbol{\delta}\\ \Delta|\boldsymbol{V}|\end{bmatrix} = \begin{bmatrix}\Delta\mathbf{P}\\ \Delta\mathbf{Q}\end{bmatrix}. \tag{4.22}$$

Here, the left matrix is the Jacobian matrix of $\mathbf{F}$. If we denote the n-th iterate by $\mathbf{x}^n$, then the next iterate is calculated as follows: $\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta\mathbf{x}$, where

$$\Delta\mathbf{x} = \begin{bmatrix}\Delta\boldsymbol{\delta}\\ \Delta|\boldsymbol{V}|\end{bmatrix}. \tag{4.23}$$

Typically, the iteration is initialized by using a *flat start*. This means that for $\mathbf{x}^0$, all voltages are set to 1 pu and all voltage angles are set to 0. When a solution of an approximate method for solving the Power Flow Problem is known, or a solution of a Power Flow Problem on a very similar network is known, it might also be used as an initial value.

5

Optimal Power Flow Problem

In this chapter, we introduce the *Optimal Power Flow problem* (OPF).

As the name suggests, optimal power flow is about finding some sort of *optimal* way to operate the grid. We have certain assets in the grid that we can control. A specification of how to control a variable of such a component is called a *set-point* (for example, a voltage magnitude set-point for a PV bus). The set-points are our decision variables[1]. Given the set-points, we can perform a power flow to get some state of the system, which needs to satisfy certain constraints. Now we want to find the set-points such that all constraints are satisfied, and do this optimally, which is, of course, characterized by some objective function. We will formulate OPF as a constrained optimization problem.

5.1. Examples of OPF Problems

The OPF problem is not a single specific problem, but a range of problems. Different real life operational scenarios come with different challenges and different available measures to tackle these challenges. When we formulate such a problem as an optimization problem, different challenges are represented by different objective functions, and different available measures are represented by different decision variables or different constraints. To illustrate this, we describe two OPF problems that relate to the two scenarios introduced in section 1.2.

A common problem in OPF literature is the *Economic Dispatch Problem*. This problem is often used as a first example of OPF, see for example [53, Chap. 8]. In this example, the active and reactive power flow demands at all loads are fixed. Each generator has a cost function that is usually a linear, piecewise linear or polynomial function of the active power supplied. The decision variables are the active power, reactive power or voltage set-points for each generator. Typically, there are bounds on the active and reactive power output of all generators, the voltage levels at all nodes and the current through each edge. In some versions of the problem, voltage set-points at PV buses are fixed, in others they are also part of the decision variables. In some cases, transformer taps and shunts could also be part of the decision variables, in others they might be fixed or not present. For TenneT, this problem is relevant in a situation where redispatch (see section 1.2.1) is required.

Another OPF problem can be formulated based on the situation described in section 1.2.2. In this scenario, all active power injections are fixed. For all loads, the reactive power injections are fixed and for a few generators, the voltage magnitude (in case of a PV generator) or the reactive power output (in case of a PQ generator) can be controlled. These generators represent the ancillary service providers. TenneT can furthermore control all transformer taps and all controllable shunts. The goal is now to minimize the cost of the ancillary services used, while keeping the voltage magnitudes within safe limits. We call this problem the *Reactive Power Dispatch problem*.

[1] The *decision variables* are the variables that can be directly controlled in the optimization process. In this case, they represent the choices that the grid operator makes.

5.2. Problem Statement

In this section, we state the OPF problem as a general (non-convex) Nonlinear Program (NLP) in the following form:

$$\begin{aligned} \underset{\mathbf{u},\mathbf{v}}{\text{minimize}} \quad & f(\mathbf{u},\mathbf{v},\mathbf{w}) \\ \text{subject to} \quad & \mathbf{g}(\mathbf{u},\mathbf{v},\mathbf{w}) = 0, \\ & \mathbf{h}(\mathbf{u},\mathbf{v},\mathbf{w}) \leq 0. \end{aligned} \tag{5.1}$$

We mostly follow [35] in this formulation. For now, we assume the decision variables are allowed to vary continuously, but in section 5.4, we will have a closer look at this assumption. We state the problem in the Polar *Power-Voltage Formulation*, first described in [17] in the 1960s [12, p. 25].

5.2.1. Variables

In (5.1), $\mathbf{u}$ is the vector of decision variables or controllable variables[2]. These can be:

- P_i, when i is a PV bus or a PQ bus
- Q_i, when i is a PQ bus
- $|V_i|$, when i is a PV bus or Slack bus
- τ_{kl} and θ_{kl} when $\{k,l\}$ is a transformer edge with controllable tap ratio
- Y_i^{ns}, when i is a node with a controllable shunt attached

$\mathbf{w}$ is the vector of exogenous variables or fixed variables. These can be:

- P_i and Q_i for each PQ-node i, when it is not a decision variable
- P_i and $|V_i|$ for each PV-node i, when it is not a decision variable
- $|V_i|$ and δ_i, for each slack node i, when it is not a decision variable
- Y_{kl}^{sr} and B_{kl}^{es} for each edge $\{k,l\}$
- Y_i^{ns} for each node i, and τ_{kl} and θ_{kl} for each edge $\{k,l\}$, when it is not a decision variable

$\mathbf{v}$ is the vector of state variables. These can be:

- $P_i, Q_i, |V_i|$ and δ_i for each node i, when it is not a decision variable or exogenous variable

As a general rule, the exogenous variables are the variables that are fixed, the decision variables are the variables that can be directly controlled, and the state variables are variables that cannot be directly controlled, but that can change if the decision variables are changed. Since $\mathbf{w}$ is fixed, we will omit it in the rest of this report. A solution of this problem is some $\mathbf{u}^{\text{opt}}$ that minimizes the objective. Note that for a given decision vector $\mathbf{u}$, we can always obtain the corresponding state $\mathbf{v}$, by performing a Power Flow calculation.

Note that the variables described under decision variables **can** be decision variables, but are not necessarily decision variables. Whether they are decision variables or not depends on the exact scenario that is modelled by the OPF problem (see section 5.1). Remember from section 3.2.1, that a bus without loads or generators attached to it, is modelled as a bus where $P_i = Q_i = 0$.

We remark that for the (ordinary) OPF problem, the distinction between decision variables and state variables is sometimes arbitrary. Take for example those nodes where one of Q_i and $|V_i|$ is a decision variable, and the other a state variable. It does not matter which of the two is the decision variable and which is the state variable. On both state variables and decision variables we can define constraints, and in the end, both are available in the OPF solution. Similarly, we also do not necessarily need a designated slack bus, as long as we have at least one node where the active power output is either a state or a decision variable [12, p. 25]. In Security Constrained OPF, however, this distinction **is** important, as we will see in section 5.3.

[2] We do not specify in what specific order these variables appear as the components of $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$, as this is not really relevant. We never talk about a component of $\mathbf{u}$, $\mathbf{v}$ or $\mathbf{w}$ at a specific index. The vectors are more meant as a shorthand for *all variables that are decision/state/exogenous variables.*

Formulation in Terms of Load and Generator Power Injection
In some cases, problem formulations state that voltage set-points and active or reactive power set-points for **generators** or **loads** are controllable, instead of stating that either $|V_i|$, P_i or Q_i is controllable for a **bus**. If all buses have at most one load or generator attached, this makes no difference. If a bus has multiple loads and generators attached, then P_i and Q_i are the net active and reactive power injections. Therefore, we may formulate the problem in terms of generator and load power injections, instead of nodal power injections.

Formulating the problem with load and generator power injections can be beneficial in situations where it is more accurate to define constraints or cost functions for single loads or generators, to accurately model real life situations. These constraints and cost functions could be combined to obtain constraints and cost functions for nodal power injections, but this is not always a trivial task, and can lead to more complex types of constraints or cost functions (e.g., two generators with a linear cost function can have a piecewise linear combined cost function).

We can transform the variables for node i in the following way:

- P_i may be substituted as follows:
$$P_i = \sum_{l \in \mathcal{L}_i} P_l + \sum_{g \in \mathcal{G}_i} P_g \tag{5.2}$$
- Q_i may be substituted as follows:
$$Q_i = \sum_{l \in \mathcal{L}_i} Q_l + \sum_{g \in \mathcal{G}_i} Q_g \tag{5.3}$$
- all synchronous generators that are connected to the same bus, must be operating at the same voltage, so no substitutions for $|V_i|$ or δ_i are made

Here, we denote the set of all generators and loads attached to node i, by $\mathcal{G}_i$ and $\mathcal{L}_i$, respectively, and P_l, P_g, Q_l and Q_g are the active and reactive power injections for a single load or generator. Here again, the power injections are exogenous variables if they are fixed, decision variables if they are controllable, and state variables otherwise.

In the rest of this chapter, we continue to use the formulation in terms of nodal power injections for simplicity of notation. We remark that when the problem is instead formulated in terms of load and generator power injections, the objective and constraints may also be functions of the load and generator power injections.

5.2.2. Equality Constraints
The equality constraints should enforce that all variables together correspond to a valid system, that is, that they should satisfy the non-linear system of equations of the power flow problem. This means that for each node k, equations (4.14) and (4.15) should be satisfied.

5.2.3. Inequality Constraints
The inequality constraints represent physical limits or safety limits on components in the grid. These can be:

- limits on the voltage magnitude at each node
- limits on active or reactive power injections at each node
- limits on current through each edge
- limits on shunt impedance or transformer tap ratios

Often these limits will be simple upper and lower bounds on one of the state or decision variables, but they can also be more complex. For example, the current through each edge is not explicitly included in the decision or state variables, but we can calculate the current flowing through each edge via equation (3.19). An upper bound on the current through an edge $\{k, l\}$ would then look as follows:

$$|Y_{kl}(V_l - V_k)| \leq I_{kl}^{\max} \tag{5.4}$$

for some number $I_{kl}^{\max}$. In Appendix B, expressions of two type of current constraints are given.

5.2.4. Objective Function

The objective function can be a cost function that needs to be minimized, for example, the cost of generators in the economic dispatch problem. In some cases, the objective might also be to minimize losses, carbon emissions or some other measure of performance. We now list some examples.

Minimization of Costs Given a cost per MVA of active or reactive power output for each generator, we can minimize the total cost. Similarly, given an amount of CO_2 emissions per MVA of active or reactive power output for each generator, we can minimize total CO_2 emissions.

Minimization of Power Losses The sum of all active power injections is always non-negative, since the power injections need to compensate for losses in the system. We can minimize the total losses in the system by minimizing $\sum_{i \in \mathcal{N}} P_i$.

Reactive Power Reserve The *reactive power reserve* of a generator is the ability of a generator to supply or consume extra reactive power compared to its operating condition. This reserve generating capacity can be used to maintain the voltage level at a bus, in case extra reactive power supply or consumption is needed due to a contingency. In [38], a *generator reactive margins* term is included in the objective, which looks as follows:

$$\sum_{g \in \mathcal{G}} \left(\frac{Q_g}{Q_g^{\max}} \right)^2 . \tag{5.5}$$

Here $\mathcal{G}$ is the set of all generators, and

$$\begin{aligned} Q_g &= \text{ reactive power output of generator } g \\ Q_g^{\max} &= \text{ maximum reactive power output of generator } g. \end{aligned}$$

By minimizing this objective, the reserve capacity to produce reactive power is maximized. The grid analysis software `PowerFactory` (see section 6.2) includes capabilities to minimize the deviation in reactive power output of generators from either the minimum, the maximum, or some target reactive power output. These are useful (in respective order) in scenarios where critical voltage drops, critical voltage rises, or both might occur [26, sec. 38.2.1.1].

5.3. Security Constrained Optimal Power Flow

In *Security Constrained Optimal Power Flow* (SCOPF), we additionally consider the behaviour of the grid under the influence of contingencies. As we mentioned in section 3.3, it is usually required that the grid is operating in an N-1 secure state, meaning that if any contingency were to occur, all constraints are still satisfied. To ensure this, we need to also look at the state that the system will go to once the contingency has occurred, and see if it still satisfies all limits.

We have noted that we can prepare for certain contingencies (namely those that lead to critical voltage drops or rises) by maximizing reactive power reserve. With this reactive power reserve, we could react to any voltage issues that might occur in case of a contingency. However, such a method does not guarantee that all voltage issues can be resolved, because we never check whether there actually is enough reserve power capacity to resolve all problems. Another issue is that set-points for generators would have to be changed quickly, if a swift restoration of voltage levels is required. This might not always be possible. Furthermore, contingencies could also lead to other problems, such as edge overloading, that cannot be resolved with extra reactive power generation capacity. This means that maximizing reactive power generation is not a complete solution for SCOPF. Instead, we formulate the SCOPF problem as another NLP.

5.3.1. Mathematical Formulation

From now on, we will denote the objective function, equality constraints, and inequality constraints of the (non-security constrained) OPF by f_0, $\mathbf{g}_0$ and $\mathbf{h}_0$, respectively. Given decision variables $\mathbf{u}_0$, we

denote the state of the system by $\mathbf{v}_0$. Once a contingency occurs, we essentially obtain a different grid, where the topology of the network might have changed, the admittance matrix might have changed, or power injections might have changed. This new grid also has different constraint functions[3], which we denote by $\mathbf{g}_c$ and $\mathbf{h}_c$ for a contingency c.

We differentiate between two kinds of SCOPF, namely, *preventive* (or *preventative*) and *corrective* SCOPF. We first introduce preventive SCOPF.

5.3.2. Preventive SCOPF

In *preventive SCOPF* we require that for every contingency c, if that contingency occurs, the state of the system still satisfies all constraints $\mathbf{g}_c$ and $\mathbf{h}_c$, with the pre-contingency decision variables $\mathbf{u}_0$. This leads to the following constraints:

$$\mathbf{g}_c(\mathbf{u}_0, \mathbf{v}_c) = 0 \qquad \text{for } c \in C \tag{5.6}$$

$$\mathbf{h}_c(\mathbf{u}_0, \mathbf{v}_c) \leq 0 \qquad \text{for } c \in C. \tag{5.7}$$

Note that here $\mathbf{u}_0$ are the same decision variables as the pre-contingency case, but that the state $\mathbf{v}_c$ does change, since a contingency will change the voltages and power injections throughout the grid. In section 5.2.1, we remarked that for ordinary OPF, in some cases, the distinction between decision and state variables is arbitrary, but for SCOPF, this is not the case. A state variable can vary across contingency states, but a decision variable can not.

We now obtain the following optimization problem:

$$\begin{aligned} \underset{\mathbf{u}_0; \mathbf{v}_0, \ldots, \mathbf{v}_C}{\text{minimize}} \quad & f(\mathbf{u}_0; \mathbf{v}_0, \ldots, \mathbf{v}_C) \\ \text{subject to} \quad & \mathbf{g}_0(\mathbf{u}_0, \mathbf{v}_0) = 0, \\ & \mathbf{h}_0(\mathbf{u}_0, \mathbf{v}_0) \leq 0, \\ & \mathbf{g}_c(\mathbf{u}_0, \mathbf{v}_c) = 0 \quad \text{for } c \in C, \\ & \mathbf{h}_c(\mathbf{u}_0, \mathbf{v}_c) \leq 0 \quad \text{for } c \in C. \end{aligned} \tag{5.8}$$

In general, the cost function can be any function of the control variables and all states. An obvious example, is to take the cost function of the N-0 state (i.e., $f_0(\mathbf{u}_0, \mathbf{v}_0)$), because the grid is most likely to operate in this state. Another example is to take a probability-weighted average of the cost function for all contingency states (i.e., $\sum_{c=0}^{C} p_c \, f_0(\mathbf{u}_0, \mathbf{v}_c)$, where p_c is the probability that contingency c occurs).

We remark that this formulation leads to a considerable increase in constraints and state variables, compared to a formulation with just the pre-contingency constraints. The number of constraints and state variables are approximately multiplied by $C + 1$. The amount of decision variables does not change. This also means that there might not always be a feasible solution, even if the problem without security constraints has a solution. If, for example, all lines are close to their maximal current rating in the pre-contingency state, and all P_i's are fixed (i.e. we do not allow redispatch), a contingency state where a line fails could be unfeasible.

5.3.3. Corrective SCOPF

In *corrective SCOPF* we allow the decision variables to vary after the occurrence of a contingency, up to a specified amount. This leads to the following optimization problem:

$$\begin{aligned} \underset{\mathbf{u}_0, \ldots \mathbf{u}_C; \mathbf{v}_0, \ldots, \mathbf{v}_C}{\text{minimize}} \quad & f(\mathbf{u}_0, \ldots, \mathbf{u}_C; \mathbf{v}_0, \ldots, \mathbf{v}_C) \\ \text{subject to} \quad & \mathbf{g}_0(\mathbf{u}_0, \mathbf{v}_0) = 0, \\ & \mathbf{h}_0(\mathbf{u}_0, \mathbf{v}_0) \leq 0, \\ & \mathbf{g}_c(\mathbf{u}_c, \mathbf{v}_c) = 0 \quad \text{for } c \in C, \\ & \mathbf{h}_c(\mathbf{u}_c, \mathbf{v}_c) \leq 0 \quad \text{for } c \in C, \\ & |\mathbf{u}_0 - \mathbf{u}_c| \leq \Delta\mathbf{u} \quad \text{for } c \in C. \end{aligned} \tag{5.9}$$

[3]Many constraints will be the same. Some constraints might no longer be included, (e.g., the contingency is the removal of a line, and therefore the constraint on the loading of that line is no longer relevant). Some constraints might have different bounds (e.g., the contingency is the removal of one of two parallel lines, and therefore the edge can only transport half of the current).

Note that the fact that the decision variables are allowed to change, is reflected in the argument $\mathbf{u}_c$ in the third and fourth constraint. The last constraint restricts by how much the decision variables are allowed to change when a contingency occurs. Some set-points might be allowed to change freely, others might only be allowed to change by a certain amount, or not at all, because of physical limitations on how quickly these set-points could be changed in the case of a contingency.

When the same objective function is used, corrective SCOPF will always find a solution with a better objective value, compared to preventive SCOPF, because the latter is a special case of corrective SCOPF with $\Delta\mathbf{u} = 0$. The number of state variables and constraints in this formulation is comparable to preventive SCOPF, however, the number of decision variables is multiplied by $C + 1$.

5.4. Continuity of Decision Variables

So far, all constraints that we have introduced were continuous. However, in real-world applications, components such as shunts and transformer taps cannot always be controlled continuously, but have a discrete set of possible set-points. This might be binary (i.e., on or off), or a finite amount of possible set-points. In this case, the optimization problem becomes a *Mixed Integer Nonlinear Program* (MINLP). This makes the problem harder to solve. When we have a (SC)OPF problem with discrete variables, its *continuous relaxation* is the problem where all constraints that restrict a single decision variable to a finite or countable set are dropped. In section 6.3.2, we have a look at some ways to deal with discrete variables.

5.5. Other Formulations of the OPF Problem

In this chapter, we have formulated the (SC)OPF problem in terms of active and reactive power and voltage angle and magnitude. This is however not the only way to formulate the problem. It can for example also be formulated with real and imaginary voltages, or with complex currents instead of complex powers [12]. The OPF problem (or slight relaxations of the problem) can also be formulated as a quadratically constrained quadratic program or a semidefinite program [6].

Most scientific literature on SCOPF uses the formulation that we used here [12, sec. 6]. Since set-points and ratings of real-life components are stated in active and reactive power, and voltage magnitude and angle (e.g., specifications for a transformer will provide a range of voltage magnitudes in which it is safe to operate, it will not provide a range for real and imaginary voltage), it is also the formulation that is most easy to apply in a real-world application.

To limit the scope of the research, we stick to the formulation that is used in this chapter.

6

Review of Existing Methods for OPF

In chapter 5, we have formulated the (Security Constrained) Optimal Power Flow problem. The aim of this chapter is to review existing methods for solving the optimal power flow problem, and identify the challenges that come with the problem. We first have a look at two existing software packages for (SC)OPF, and then give an overview of the scientific literature on the problem.

6.1. pandapower

`pandapower` [50] is an open-source software package for the analysis and optimization of power systems. It is entirely written in `Python`, it makes use of `pandas` [49], and it builds upon methods and implementations from `PYPOWER` [36]. `pandapower` includes a way to formulate certain types of OPF problems, and can interface with methods from `PYPOWER` and `PowerModels.jl` [19] for solving these OPF problems. The specifics of the formulation are as follows:

- active and reactive power injections of loads and PQ generators are controllable
- active power injections of synchronous generators are controllable
- transformers and shunts are not controllable
- cost functions are quadratic polynomials or piecewise linear functions of P_i and Q_i for all controllable components
- upper and lower bounds can be defined on P_i and Q_i for all controllable assets and $|V_i|$ for all buses
- an upper limit can be defined on $|I_{kl}|$ for all edges

We can see that this formulation lacks the ability to have voltage magnitudes as decision variables, however, as we saw in section 5.2.1, there is a workaround for this, by modelling all PV buses as PQ buses. The formulation also lacks control for more advanced components such as shunts or transformers.

When the `PYPOWER` solver is called, it makes use of a primal-dual interior point method, based on the one described in [52]. When the `PowerModels.jl` solver is called, it uses the `Ipopt` [51] solver by default, but it can also use other solvers, such as those of `Gurobi` [28].

6.2. PowerFactory

`PowerFactory` is a commercial software application for modelling and analysis of power systems. It includes an extensive set of network analysis tools, including Power Flow and Optimal Power Flow capabilities. `PowerFactory` is licensed by DIgSILENT GmbH.

The OPF module of `PowerFactory` does not distinguish PV generators and PQ generators, and only allows active and reactive power output as decision variables (see section 5.2.1 on why this can be done). We give an overview of the OPF capabilities of `PowerFactory`:

- active and reactive power output of generators are controllable
- transformer tap positions are controllable (continuous or discrete)
- shunt admittances are controllable (continuous or discrete)
- the following types of objectives can be chosen: minimization of losses, minimization of costs, minimization of load shedding, maximization of reactive power reserve, minimization of control variable deviations
- upper and lower bounds can be set on generator active and reactive power injections and $|V_i|$ for all buses
- an upper limit can be defined on $|I_{kl}|$ for all edges

`PowerFactory` also provides a solver for a simplified version of the model, based on the DC load flow method. In this formulation, a linear approximation is made to the equality constraints. This version is an (Integer) Linear Programming (ILP) formulation and is solved using the simplex method and a branch-and-bound algorithm [26, sec. 25.2.1]. With this simplified model, it is also capable of doing SCOPF.

`PowerFactory` is an application developed for Microsoft Windows, primarily designed to be run on desktop computers. It has an extensive graphical interface, making it suitable for obtaining an overview of large grid models and easy editing. It also has an application programming interface (API) in the programming language `Python`, through which grid data, set-points and results of calculations can be imported and exported.

6.2.1. Downsides of Closed Source Nature of PowerFactory

Since it is not possible to view or modify the source code of `PowerFactory`, it is not as flexible as open-source software. We list some specific issues that TenneT has, that prevent the usage of `PowerFactory` in some parts of its operation.

- Constraints and cost functions for OPF calculations have to be chosen from a specific list and have to be provided in a specific form. This limits the types of OPF calculations that can be performed with the software.
- Performing bulk calculations and working with time series in `PowerFactory` is slow and convoluted. It is not straightforward to do bulk calculations where the grid configuration and power injection scenarios are varied independently.
- There is no access to the details of internal algorithms and the output of intermediate statuses and results in iterative algorithms is limited. This makes it less suited for scientific research because it is hard to make comparisons with other methods.
- When algorithms in `PowerFactory` do not converge, the lack of output and insight into the internal workings of algorithms makes it harder to find the cause of the convergence issues.
- Any additional functionality has to be requested from the manufacturer. There is no guarantee that these features will be implemented, and when they will be available.
- The focus by the manufacturer on desktop use, makes it harder to integrate the software as part of server-based toolchains or automated processes. It does not support the usage on Linux based operating systems, for example.

6.3. Scientific Literature

In this section, we do a literature study on the OPF problem, focussing on three topics: reducing computational complexity, handling of discrete variables and use of (MI)NLP solvers. We restrict our attention to literature that uses a similar formulation of the OPF problem as we do.

6.3.1. Reducing Computational Complexity

When performing OPF calculations close to real time, or when performing large amounts of OPF calculations, calculation time can be an important factor in the applicability of an OPF method. For SCOPF, calculation time is an even bigger factor, as the computational cost of SCOPF calculations grows

rapidly with an increase in the amount of considered contingencies. We quote from [16, sec. 4.1.1]: "The major challenge of the SCOPF is the size of the problem, especially for large systems and [cases] where many contingencies are considered. Trying to solve this problem directly for a large power system, by imposing simultaneously all post-contingency constraints, would lead to prohibitive memory and CPU times requirements."

In the literature, a few methods are suggested to reduce the computational complexity. A first approach is linearization of the problem. This approach can be used to reduce the computational complexity of general OPF problems. For SCOPF problems specifically, three other common approaches can be observed: contingency filtering, problem decomposition, and network compression [16, sec. 4.1].

Linearization of the Problem
Since linear programs are, in general, much easier to solve than nonlinear programs, some literature suggests transforming the OPF problem into a linear program. Linearizing the OPF problem involves finding a linear approximation for the cost function f, and the constraint functions $\mathbf{g}$ and $\mathbf{h}$. In [2, sec. 4], two methods for linearizing $\mathbf{g}$ are suggested, one simply linearizes g by use of a first order Taylor approximation, the other method (which is used more often [47, sec. II.C]) does this too, but makes some additional assumptions, leading to a simpler linear system. The second method shares much of the assumptions of the DC power flow method, and is therefore referred to as the DC OPF problem. The linear approach is, however, less suited for situations where accurate modelling of reactive power is important, since the behaviour of reactive power tends to be more nonlinear [47, 13].

Methods for SCOPF
A first approach for reducing the computational complexity of SCOPF is called contingency filtering. The idea behind this approach is to only consider *binding constraints*, that is, those constraints that would lead to a better objective value if they were dropped. This is not known beforehand, but contingency filtering techniques exist, that select contingencies that are likely to have binding constraints. Examples of such techniques can be found in [22, 31, 42].

A second approach is based on the *generalized Benders Decomposition*. In this decomposition, the problem is split into one master problem and a sub-problem for each contingency. In each sub-problem, binding constraints are identified, and these are then incorporated in the master problem via a so-called *Benders Cut*. This process greatly reduces the computational complexity, and it allows for parallelization of certain parts of the computation, leading to an additional speedup. The downside is that on non-convex problems such as the SCOPF problem, the Benders Decomposition solves an approximation of the original problem, meaning that an obtained solution is not guaranteed to converge to an optimal solution of the original SCOPF problem. This method has been described in [39, 35].

Another method to reduce the computational complexity of the SCOPF problem is the one presented in [31]. It makes use of the fact that the effect of a contingency is usually only felt in a region around the contingency. For each contingency, an active region is identified. This is the region where the contingency has a significant impact on voltages and power flows. They then replace all nodes outside the active region with a so-called *REI-DIMO equivalent network*. This means that the nodes outside the active region are replaced by a reduced number of equivalent nodes, greatly reducing the size of the constraints belonging to that contingency. In numerical experiments, a solution is obtained using this method for a very large model (9241 buses, 12000 contingencies) of the European grid, while having some method to deal with discrete variables, in just over an hour of calculation time [42].

6.3.2. Handling of Discrete Variables
Both in ordinary OPF and SCOPF, discrete variables make the problem significantly harder. For very large SCOPF problems, classical MINLP methods such as branch and bound are incapable of obtaining solutions in reasonable time frames [16, sec. 4.3]. Another complication is that some state-of-the-art MINLP solvers do not support trigonometric functions [1]. In [23, sec. IV], it is mentioned that solving the OPF and SCOPF problem directly with MINLP solver `Bonmin` was attempted for a grid with 60 nodes and 33 contingencies. The solver found a solution for the OPF problem, but did not provide a feasible solution for the SCOPF problem after several hours of running. Instead of MINLP methods, some algorithms use continuous relaxations of the problem, and then use rounding techniques to obtain

solutions that are feasible for the discrete problem. These techniques cannot guarantee optimality of the solution, however.

A simple rounding method for mixed integer OPF problems is described in [34, sec. 2.1]. First, a solution of the continuous relaxation of the OPF is obtained. Then all variables that are required to be discrete, are rounded to their nearest discrete value. The OPF problem is then solved again with the discrete variables fixed to their rounded off values, the continuous decision variables are now the only decision variables.

In [32, sec. II.D] a progressive round off method is suggested for SCOPF. This method first rounds off those variables that are nearest to a discrete value. Then another round of optimization is done with the rounded off variables fixed, and all other variables still considered as continuous. Then, again, the variables that are closest to a discrete value are rounded off. This is iterated until all values are rounded off. This method is also used in [42].

Another method is introduced in [34, sec. 3.3], the so-called *objective feasibility pump*. Once again, first a solution is obtained of the continuous relaxation. Then, for each variable that is required to be discrete, the distance of that variable to its closest discrete value is added to the objective as a penalty term. These terms are called the *feasibility pump*. Now an iterative process starts, where in each iteration the OPF with feasibility pump term added is solved, and after each iteration the weight of the feasibility term is increased. The process halts when a solution is obtained that satisfies all discrete constraints (up to a certain tolerance).

The method proposed in [23] uses a similar approach as the objective feasibility pump. First, it converts every discrete variable into a set of binary variables. Then it relaxes the problem to a continuous problem and adds a penalty term to the objective for every binary variable. The penalty term is based on the *Fischer Burmeister function* and it penalizes the binary values for being away from zero or one. In numerical experiments [23, sec. IV], the proposed method outperforms the simple rounding method, both in finding feasible solutions and in finding more optimal solutions.

A downside to the latter three methods is that they all have one or multiple parameters that require tuning. For the progressive round off method this is the bin size of which variables are considered close enough to be rounded off each iteration. In the object feasibility pump method it is the initial weight of the penalty terms. In the last method, it is a parameter for the Fischer Burmeister function and a weight for the penalty terms.

6.3.3. (MI)NLP Solvers and Algorithms

The challenge of finding solutions of (MI)NLP problems is not exclusive to optimal power flow. Therefore, a lot of literature makes use of existing solvers, or algorithms proposed in other literature, to obtain solutions. These are some of the solvers that can be found in the literature:

- [22, 34, 23] use the `Ipopt` solver [51], an open-source software package for large-scale nonlinear optimization. It implements an interior point line search filter method. `PowerFactory` and `pandapower` also have the option to use `Ipopt`.
- The `PYPOWER` OPF solver is a primal-dual interior point method based on [52].
- [42] makes use of a solver based on [10], a primal-dual interior point method using projected conjugate gradient iteration.
- [32] uses the commercial solver `Knitro` [11], which implements multiple algorithms, the two relevant ones are both primal-dual interior point methods, one of them is based on [10].
- [35] uses a `MATLAB` solver.
- [14] refers to [15], which compares three interior point methods for OPF problems. It recommends a *predictor-corrector* and a *multiple centrality corrections* based interior-point method.
- [23] mentions that the `Bonmin` [20] solver is able to solve non-security constrained OPF problems, but is not able to solve SCOPF problems. `Bonmin` is an experimental open-source MINLP solver for general MINLP problems.

6.3.4. Optimality of Solutions

The OPF problem is in general a non-convex, non-linear optimization problem. This means that local minima might exist. In fact, these have been shown to exist [8]. Not many claims are made about the optimality of the solutions that different methods obtain. Some claims are made about local optimality for continuous relaxations of the SCOPF problem [42], but to the author's knowledge, no methods claim to be able to find globally optimal solutions, for all but very small-scale problems. We quote from [48, sec. 2.3]: "It is unlikely that useful theories of convergence or global optimality can be developed for non-trivial real-life OPF problem formulations." Instead, the focus lies on reliably obtaining feasible solutions, with acceptable objective values.

6.4. Conclusion

Both in literature and in commercial software, methods have been suggested and implemented to perform (SC)OPF. It is, however, still an active area of research and no standard method exists that is both fast and accurate and always finds globally optimal solutions for security constrained OPF with discrete decision variables.

There are existing open-source methods, such as `pandapower`, but the capabilities and configurability of these methods is quite limited. Commercial solutions such as `PowerFactory` have more capable (SC)OPF methods, but their closed-source nature restricts their applicability to a specific set of OPF problem types because cost functions and constraints can only be configured in a particular form. Furthermore, implementing (SC)OPF calculations as part of an automated process, or as part of bulk calculations, where (SC)OPF calculations are performed on large amounts of different scenarios and grid variations, can be difficult with commercial solutions, since the configuration options of the software can be limited.

From the scientific literature, it is clear that solving large scale (SC)OPF problems has two major challenges: high computational costs, and treatment of discrete variables. On both of these challenges, a lot of research has been done, and some methods for dealing with these challenges have been suggested. Although some comparisons have been made between different methods, there is no wide-spread consensus in the literature about which of these methods perform the best. In most cases, the methods involve an iteration process for checking contingency states, dealing with violated constraints in contingency states, possibly dealing with discrete variables and obtaining a new solution. Part of this iteration process is also solving a (non-)linear program, which is often done using interior point methods, either via existing solvers or by implementing algorithms found in the literature. Some methods in the literature are capable of doing large scale SCOPF in reasonable timeframes, with some way of dealing with discrete variables. In general, no claims are made about the global optimality of solutions obtained by any of the methods.

7

Methodology

In Chapter 6, various methods for OPF have been presented. For our implementation, we have opted for a *primal-dual interior-point algorithm*, implemented in the open-source software package `Ipopt` [51]. We implement the model as described in Chapter 5 in `Python` making use of the optimization modelling framework `Pyomo` [9, 29]. In this chapter we describe the details of the implementation, and give a brief introduction to the primal-dual interior-point method implemented by `Ipopt`.

7.1. Ipopt: a primal-dual interior point solver

To obtain a solution for the OPF problem, we use a primal-dual interior point algorithm with filter line-search, as implemented in `Ipopt`[1] [51], an open-source software package for nonlinear optimization. The full algorithm is described in [51], but in this section we highlight some of its steps. For an introduction to primal-dual interior point methods, we refer to [7, 40].

7.1.1. Introducing Slack Variables

The algorithm we describe can treat optimization problems of the following form:

$$\begin{aligned} \underset{\mathbf{x},\,\mathbf{s}}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{g}(\mathbf{x}) = \mathbf{0}, \\ & \mathbf{h}(\mathbf{x}) + \mathbf{s} = \mathbf{0}, \\ & \mathbf{s} \geq \mathbf{0}, \end{aligned} \tag{7.1}$$

where all functions and variables are real, and all functions are twice continuously differentiable. The OPF problem, that we formulated in Section 5.2 is not in this form. However, we can add slack variables to bring into this form.

We combine the decision variables $\mathbf{u}$ and the state variables $\mathbf{v}$ that were introduced in Section 5.2.1 into a single vector $\mathbf{x}$, which we call the vector of *optimization variables*. The exogenous variables are constant and are therefore not treated as variables, but as parameters of the problem. We denote the number of optimization variables by n_1, and we write $\mathbf{x} = (x_1, \ldots, x_{n_1})$.

Both constraint functios $\mathbf{g}$ and $\mathbf{h}$ are functions of the decision variables and state variables, so they can be written as a function of $\mathbf{x}$. For every inequality constraint, of the form $h_i(\mathbf{x}) \leq 0$, we introduce a slack variable s_i, and replace the constraint by two new constraints: $h_i(\mathbf{x}) + s_i = 0$ and $s_i \geq 0$. These two constraints are equivalent to the original constraint. The slack variables make up the vector $\mathbf{s}$. The problem is now written in the form of (7.20), and it is equivalent to problem (5.1).

We denote the total number of equality constraints in the formulation by m_1, the number of inequality constraints by m_2, and we write $m = m_1 + m_2$. The number of slack variables is also equal to m_2. The total number of variables (i.e. optimization variables and slack variables) is denoted by $n = n_1 + m_2$.

[1] All experiments are performed with Ipopt version 3.14.17.

We claim that $m < n$. Since $m = m_1 + m_2$ and $n = n_1 + m_2$, this is the case if and only if the number of optimization variables is bigger than the number of equality constraints. In OPF we have 2 equality constraints per node, so we require at least $2N$ optimization variables. In chapter 4 we saw that the number of state variables (which are also optimization variables) is $2N$, so if there is at least 1 decision variable, we indeed have $m < n$.

7.1.2. Lagrangian and Modified KKT Equations

In this section, we formulate the Lagrange dual function (or Lagrangian), the barrier problem and the modified KKT conditions of the problem. For an introduction to these topics, we refer to [7] and [40].

The Lagrangian of this problem is $L(\mathbf{x}, \mathbf{s}; \boldsymbol{\lambda}, \mathbf{z}) = f(\mathbf{x}) + \boldsymbol{\lambda}^\mathsf{T}\mathbf{g}(\mathbf{x}) + \mathbf{z}^\mathsf{T}(\mathbf{h}(\mathbf{x}) + \mathbf{s})$, with dual variables $\boldsymbol{\lambda} \in \mathbb{R}^{m_1}$, and $\mathbf{z} \in \mathbb{R}^{m_2}$.

This problem has the following log-barrier problem, for a barrier parameter μ:

$$\begin{aligned} \underset{\mathbf{x},\,\mathbf{s}}{\text{minimize}} \quad & \phi_\mu(\mathbf{x}, \mathbf{s}) = f(\mathbf{x}) - \mu \sum_{i=1}^{m_2} \log(s_i) \\ \text{subject to} \quad & \mathbf{g}(\mathbf{x}) = \mathbf{0}, \\ & \mathbf{h}(\mathbf{x}) + \mathbf{s} = \mathbf{0} \end{aligned} \tag{7.2}$$

The idea of the log-barrier problem is that optimal solutions $\mathbf{x}^\star(\mu)$ for (7.2) approach the optimal solution of the original problem (7.20) as μ goes to 0. We can therefore find a solution to our orignal problem by solving the barrier problem for smaller and smaller μ. To solve the barrier problem we consider its Lagrangian and first order optimality conditions.

The barrier problem has the following Lagrangian:

$$\tilde{L}_\mu(\mathbf{x}, \mathbf{s}; \boldsymbol{\lambda}, \mathbf{z}) = f(\mathbf{x}) - \mu \sum_{i=m_1+1}^{m} \log(s_i) + \sum_{i=1}^{m_1} \lambda_i g_i(\mathbf{x}) + \sum_{i=1}^{m_2} z_i \left(h_i(\mathbf{x}) - \mathbf{s}\right). \tag{7.3}$$

The gradients of $\tilde{L}_\mu$ with respect to $\mathbf{x}$ and $\mathbf{s}$ are the following:

$$\nabla_\mathbf{x} \tilde{L}_\mu(\mathbf{x}, \mathbf{s}; \boldsymbol{\lambda}, \mathbf{z}) = \nabla_\mathbf{x} f(\mathbf{x}) + \sum_{i=1}^{m_1} \lambda_i \nabla_\mathbf{x} g_i(\mathbf{x}) + \sum_{i=1}^{m_2} z_i \nabla_\mathbf{x} h_i(\mathbf{x}) \tag{7.4}$$

$$\nabla_\mathbf{s} \tilde{L}_\mu(\mathbf{x}, \mathbf{s}; \boldsymbol{\lambda}, \mathbf{z}) = -\mu \sum_{i=1}^{m_2} \frac{1}{s_i} \mathbf{e}_i + \sum_{i=1}^{m_2} z_i \mathbf{e}_i, \tag{7.5}$$

where $\mathbf{e}_i$ denotes the i-th standard basis vector.

The first order optimality conditions for 7.2 are then:

$$\nabla_\mathbf{x} f(\mathbf{x}) + \sum_{i=1}^{m_1} \lambda_i \nabla_\mathbf{x} g_i(\mathbf{x}) + \sum_{i=1}^{m_2} z_i \nabla_\mathbf{x} h_i(\mathbf{x}) = \mathbf{0} \tag{7.6}$$

$$s_i z_i - \mu = 0 \quad \text{for } i = m_1 + 1, \ldots, m \tag{7.7}$$

$$\mathbf{g}(\mathbf{x}) = \mathbf{0} \tag{7.8}$$

$$\mathbf{h}(\mathbf{x}) + \mathbf{s} = \mathbf{0} \tag{7.9}$$

Here (7.6) and (7.7) are obtained by setting (7.4) and (7.5) to 0.

These are the first order optimality conditions for the barrier problem, meaning that, under some conditions on the constraints, a solution $\mathbf{x}$ of the barrier problem with parameter μ, is a local optimum, only if there exist some $\boldsymbol{\lambda}$ such that the equations (7.6) to (7.9) are satisfied[2]. Therefore, we search for a solution of the barrier problem, by applying the Newton-Raphson algorithm to the equations (7.6) to (7.9).

[2] For a more precise statement, we refer to [40, Thm. 12.1].

At each iteration, this involves solving the following system for the Newton step $(\Delta\mathbf{x}^{(k)}, \Delta\mathbf{s}^{(k)}, \Delta\boldsymbol{\lambda}^{(k)}, \Delta\mathbf{z}^{(k)})$, given a current iterate $(\mathbf{x}^{(k)}, \mathbf{s}^{(k)}, \boldsymbol{\lambda}^{(k)}, \mathbf{z}^{(k)})$ and the current barrier parameter μ:

$$\begin{bmatrix} W^{(k)} & 0 & A_{\mathbf{g}}^{(k)} & A_{\mathbf{h}}^{(k)} \\ 0 & Z & 0 & S \\ (A_{\mathbf{g}}^{(k)})^\mathsf{T} & 0 & 0 & 0 \\ (A_{\mathbf{h}}^{(k)})^\mathsf{T} & I & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x}^{(k)} \\ \Delta\mathbf{s}^{(k)} \\ \Delta\boldsymbol{\lambda}^{(k)} \\ \Delta\mathbf{z}^{(k)} \end{bmatrix} = - \begin{bmatrix} \nabla_{\mathbf{x}} f(\mathbf{x}) + \sum_{i=1}^{m_1} \lambda_i \nabla_{\mathbf{x}} g_i(\mathbf{x}) + \sum_{i=1}^{m_2} z_i \nabla_{\mathbf{x}} h_i(\mathbf{x}) \\ S\mathbf{z} - \mu\mathbf{e} \\ \mathbf{g}(\mathbf{x}) \\ \mathbf{h}(\mathbf{x}) + \mathbf{s} \end{bmatrix}, \tag{7.10}$$

where

$$\begin{aligned} \mathbf{e} &= (1, \ldots, 1) \\ X^{(k)} &= \operatorname{diag}(\mathbf{x}^{(k)}) \\ S^{(k)} &= \operatorname{diag}(\mathbf{s}^{(k)}) \\ Z^{(k)} &= \operatorname{diag}(\mathbf{z}^{(k)}) \\ A_{\mathbf{g}}^{(k)} &= J_{\mathbf{g}}(\mathbf{x}^{(k)})^\mathsf{T} = \text{Transpose of Jacobian of } \mathbf{g} \text{ at } \mathbf{x}^{(k)} \\ A_{\mathbf{h}}^{(k)} &= J_{\mathbf{h}}(\mathbf{x}^{(k)})^\mathsf{T} = \text{Transpose of Jacobian of } \mathbf{h} \text{ at } \mathbf{x}^{(k)} \\ W^{(k)} &= \nabla_{\mathbf{x}}^2 L(\mathbf{x}, \mathbf{s}; \boldsymbol{\lambda}, \mathbf{z}) = \text{ Hessian of } L \text{ w.r.t. } \mathbf{x} \text{ at } (\mathbf{x}^{(k)}, \mathbf{s}^{(k)}, \boldsymbol{\lambda}^{(k)}, \mathbf{z}^{(k)}). \end{aligned}$$

Here we used that $\nabla_{\mathbf{x}}^2 L = \nabla_{\mathbf{x}}^2 \tilde{L}_\mu$.

Obtaining a solution of this system is one of the crucial steps in the algorithm, and in section 7.1.3 we have a closer look at how this system is solved.

We do not continue the Newton steps until we have found an optimal solution for the barrier problem for a fixed value of μ, but instead we continuously update the barrier parameter, driving it to 0.

Since the OPF problem is non-convex, a point satisfying equations (7.6) to (**??**) is not guaranteed to be a (local) minimum of the barrier problem, but it could also be a maximum or a stationary point. However, under certain conditions, we can guarantee that the algorithm iterates towards a minimum. We briefly touch on these conditions in section 7.1.3, but a more detailed explanation can be found in [40, sec. 19.3.4].

The next step is to determine step sizes $\alpha^{(k)}$ and $\alpha_{\mathbf{z}}^{(k)}$ and calculate the next iterate as follows:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)} \Delta\mathbf{x}^{(k)} \tag{7.11}$$

$$\mathbf{s}^{(k+1)} = \mathbf{s}^{(k)} + \alpha^{(k)} \Delta\mathbf{s}^{(k)} \tag{7.12}$$

$$\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} + \alpha^{(k)} \Delta\boldsymbol{\lambda}^{(k)} \tag{7.13}$$

$$\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)} + \alpha_{\mathbf{z}}^{(k)} \Delta\mathbf{z}^{(k)}. \tag{7.14}$$

The step sizes are determined with a so called *line-search filter method*, which tries to find a step size such that both the objective and the constraint violation are minimized. The details of this method can be found in [51, sec. 2.3].

The algorithm requires a first iterate $(\mathbf{x}^{(0)}, \boldsymbol{\lambda}^{(0)}, \mathbf{z}^{(0)})$. An initial guess $\mathbf{x}^{(0)}$ needs to be provided by the user. The vectors $\boldsymbol{\lambda}^{(0)}$ and $\mathbf{z}^{(0)}$ can also be provided by the user, or they are calculated via the initialization procedure, which is described in [51, sec. 3.6]. In Section 7.3 we describe which initial guess we provide.

This process keeps iterating, until the iterate $(\mathbf{x}^{(k)}, \boldsymbol{\lambda}^{(k)}, \mathbf{z}^{(k)})$ satisfies equations (7.6) to (7.9), with $\mu = 0$, up to a specified tolerance.

7.1.3. Solving for the Iteration Step

A crucial step of the `Ipopt` algorithm is obtaining a solution for the linear system (7.10). This is done, equivalently, by solving the following system that is symmetric:

$$\begin{bmatrix} W & 0 & A_{\mathbf{g}} & A_{\mathbf{h}} \\ 0 & S^{-1}Z & 0 & I \\ (A_{\mathbf{g}})^{\mathsf{T}} & 0 & 0 & 0 \\ (A_{\mathbf{h}})^{\mathsf{T}} & I & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{s} \\ \Delta\boldsymbol{\lambda} \\ \Delta\mathbf{z} \end{bmatrix} = - \begin{bmatrix} \nabla_{\mathbf{x}} f(\mathbf{x}) + \sum_{i=1}^{m_1} \lambda_i \nabla_{\mathbf{x}} g_i(\mathbf{x}) + \sum_{i=1}^{m_2} z_i \nabla_{\mathbf{x}} h_i(\mathbf{x}) \\ \mathbf{z} - \mu S^{-1}\mathbf{e} \\ \mathbf{g}(\mathbf{x}) \\ \mathbf{h}(\mathbf{x}) + \mathbf{s} \end{bmatrix}, \tag{7.15}$$

where $\Sigma = S^{-1}Z$. Here, we fix k and drop the superscript $\cdot^{(k)}$ that indicates the iteration number, to simplify notation. To further simplify the notation, we define the following matrix

$$A = \begin{bmatrix} A_{\mathbf{g}} & A_{\mathbf{h}} \\ 0 & I \end{bmatrix}, \tag{7.16}$$

and we extend the matrices W and Σ with zeros, so that they are both $n \times n$ matrices:

$$W = \begin{bmatrix} \nabla^2_{\mathbf{x}} L(\mathbf{x}, \mathbf{s}; \boldsymbol{\lambda}, \mathbf{z}) & 0 \\ 0 & 0 \end{bmatrix} \tag{7.17}$$

$$\Sigma = \begin{bmatrix} 0 & 0 \\ 0 & S^{-1}Z \end{bmatrix}. \tag{7.18}$$

The matrix in (7.15) then becomes:

$$\begin{bmatrix} W + \Sigma & A \\ A^{\mathsf{T}} & 0 \end{bmatrix} \tag{7.19}$$

We call the linear system in (7.15) the *iteration system* and call the matrix (7.19) the *iteration matrix*. It is indeed a symmetric matrix: the upper right and lower left blocks are each other's transpose, and the upper left block is the sum of two symmetric matrices. The first, W, is the Hessian of a twice continuously differentiable function (assuming that f, $\mathbf{g}$, and $\mathbf{h}$ are twice continuously differentiable). The second, Σ, is the product of two diagonal matrices, and therefore diagonal.

This iteration system is solved using a direct solver for sparse symmetric matrices, such as `MUMPS` [3, 4], or `MA27` from the HSL collection [30]. When the iteration matrix is not singular, the system has a solution. This is the case when A has full rank (i.e., $\operatorname{rank}(A) = n$) and the matrix $W + \Sigma$ is positive definite on the null space of A^{T} (i.e., if for all $\mathbf{v}$ such that $A^{\mathsf{T}}\mathbf{v} = 0$, we have $\mathbf{v}^{\mathsf{T}}(W + \Sigma)\mathbf{v} > 0$) [40, Lemma 16.1].

There are two situations in which solving the system becomes problematic. The first is when the iteration matrix is singular, or very ill-conditioned. In that case, either the system doesn't have a solution, or errors induced due to finite precision arithmetic become very large, leading to inaccurate Newton steps. The second case, is when $W + \Sigma$ is not positive definite on the null space of A^{T}. In that case, it cannot be guaranteed that the Newton step obtained from the system is moving towards a minimum. The step might instead be towards a maximum, or a stationary point that is neither a maximum nor a minimum [40, sec. 19.3.4].

Both of the problematic situations can be detected by looking at number of positive, negative and zero eigenvalues of the matrix. The linear solvers that are used to solve the iteration system, also produce the number of positive, negative and zero eigenvalues. When either of the problematic situations arise, the iteration matrix is modified, by adding a diagonal matrix to the iteration matrix and trying to obtain a solution to the modified problem. The solution of this modified problem is then used as the Newton step. Details of the modification can be found in [51, sec. 3.1].

In some cases, the iteration matrix remains very ill-conditioned, even after modification. In that case, the so-called *restoration phase* is entered, in which the algorithm first tries to move to a more feasible solution, before continuing with the regular Newton steps. The details of the restoration phase can be found in [51, sec. 3.3].

A particular situation in which very ill-conditioned matrices can occur is studied in Section 8.2.

7.1.4. Scaling of OPF Problems

We have noticed that a very ill-conditioned iteration matrix can lead to problems. This can be remedied with so-called *scaling methods*. These are methods that modify the iteration system into an equivalent system that is better conditioned. We differentiate between two scaling methods: scaling of the NLP, and scaling of the iteration system.

NLP Scaling

Instead of solving problem (7.20), we can equivalently solve the following problem:

$$\begin{aligned} \underset{\tilde{\mathbf{x}}}{\text{minimize}} \quad & d_f f(D_{\mathbf{x}}\tilde{\mathbf{x}}) \\ \text{subject to} \quad & D_{\mathbf{g}}\mathbf{g}(D_{\mathbf{x}}\tilde{\mathbf{x}}) = \mathbf{0}, \\ & D_{\mathbf{h}}\mathbf{h}(D_{\mathbf{x}}\tilde{\mathbf{x}}) + \mathbf{s} = \mathbf{0}, \\ & \mathbf{s} \geq \mathbf{0}, \end{aligned} \tag{7.20}$$

$$\begin{aligned} \underset{\tilde{\mathbf{x}}}{\text{minimize}} \quad & d_f f(D_{\mathbf{x}}\tilde{\mathbf{x}}) \\ \text{subject to} \quad & D_{\mathbf{c}}\mathbf{c}(D_{\mathbf{x}}\tilde{\mathbf{x}}) = \mathbf{0}, \\ & \tilde{\mathbf{x}} \geq \mathbf{0}. \end{aligned} \tag{7.21}$$

for some $d_f > 0$ and diagonal positive definite matrices $D_{\mathbf{x}}$, $D_{\mathbf{g}}$ and $D_{\mathbf{h}}$. Then, once an optimal solution $\tilde{\mathbf{x}}^*$ is obtained for (7.21), we can multiply the solution by $D_{\mathbf{x}}$ to obtain the optimal solution for our original problem.

If we denote the blocks in the iteration matrix of the scaled problem by $\tilde{A}$ and $\tilde{W}$, then by applying the chain rule we get:

$$\tilde{A}_{\mathbf{g}} = D_{\mathbf{g}} A_{\mathbf{g}} D_{\mathbf{x}} \tag{7.22}$$

$$\tilde{A}_{\mathbf{h}} = D_{\mathbf{h}} A_{\mathbf{h}} D_{\mathbf{x}} \tag{7.23}$$

$$\tilde{W} = \left(d_f \nabla^2_{\mathbf{x}} f(D_{\mathbf{x}}\mathbf{x}) + D_{\mathbf{g}} \nabla^2_{\mathbf{x}} \boldsymbol{\lambda}^{\mathsf{T}} \mathbf{g}(D_{\mathbf{x}}\mathbf{x}) + D_{\mathbf{h}} \nabla^2_{\mathbf{x}} \mathbf{z}^{\mathsf{T}} \mathbf{h}(D_{\mathbf{x}}\mathbf{x})\right) D^2_{\mathbf{x}}. \tag{7.24}$$

This means that the columns of $\tilde{A}$ and $\tilde{W}$ are scaled by the diagonal elements of $D_{\mathbf{x}}$ and $D^2_{\mathbf{x}}$, respectively. Furthermore, $D_{\mathbf{g}}$ and $D_{\mathbf{h}}$ scale the rows of $\tilde{A}_{\mathbf{g}}$ and $\tilde{A}_{\mathbf{h}}$ and the constraint terms of $\tilde{W}$, and d_f scales the objective term of $\tilde{W}$. The right-hand side of the iteration system scales similarly. The Newton step obtained at each step of the algorithm is invariant under the scaling (when we do not consider floating point inaccuracies) [51, sec. 3.8], however, the magnitude of the components of the iteration matrix and the condition number of the iteration matrix are not invariant under the scaling. We could therefore choose $D_{\mathbf{x}}$, $D_{\mathbf{g}}$, $D_{\mathbf{h}}$ and d_f in such a way that the condition number of the iteration matrix is minimized. The implementation of `Ipopt` uses a rather conservative scaling by default, only scaling the objective and constraint functions when the first partial derivatives become very big [51, sec. 3.8].

Iteration System Scaling

Instead of scaling the whole problem, the iteration system could also be scaled. This is a common method in the solution of linear systems, and is also known as *matrix equilibration*. In this method, instead of solving the system $M\mathbf{v} = \mathbf{b}$ for some matrix M and vector $\mathbf{b}$ (e.g., the iteration system), the equivalent system $(D_1^{-1} M D_2)\mathbf{w} = D_1^{-1}\mathbf{w}$ is solved, and then obtaining $\mathbf{v}$ via $\mathbf{v} = D_2\mathbf{w}$. Here, once again, D_1 and D_2 are diagonal positive definite matrices, that are picked such that the condition number of $D_1^{-1} M D_2$ is minimal. Finding D_1 and D_2 such that the condition number is exactly minimal is difficult, so usually a fast approximate method is used [27, sec. 3.5.2].

A major difference with the NLP scaling method, is that the iteration system scaling is performed every iteration, as opposed to just once, at the beginning of the algorithm. Depending on which linear solver is chosen, `Ipopt` will perform linear system scaling with one of a few different methods, such as the `MC19` procedure or the `MC64` procedure, both part of the HSL collection [30].

7.2. Implementation of the OPF Problem in Pyomo

In Section 7.1 we have noted that, to solve the OPF problem, we need a way to evaluate the objective function, constraint functions and its first and second derivatives with respect to the optimization variables. For this, we use `Pyomo` [9, 29]. This is an open-source package for modelling optimization problems in the programming language `Python`. It provides classes for concepts commonly used in the formulation of optimization problems, such as index sets, variables, parameters, constraints and objective functions. It therefore allows for a relatively straightforward conversion from a purely mathematical formulation of a problem, to an object in `Python`, representing this problem. `Pyomo` can then interface with one of many solvers, to obtain a solution to the problem. `Ipopt` is one of those solvers.

Implementing our OPF problem in `Pyomo` is straightforward. First the necessary variables need to be defined. Then, to define the objective and constraint functions, an expression for each function needs to be provided, in terms of the previously defined variables. An initial guess is provided by giving a value to all variables. To solve the problem, an executable file of the solver needs to be provided, along with any desired solver options. `Pyomo` will then handle all interactions with the solver and if the solver indicates that it has found an optimal solution, it will provide the optimal solution to the user.

To calculate first and second derivatives of the objective and constraint functions, we use automatic differentiation. `Pyomo` interfaces with `Ipopt` through the AMPL Solver Library [25], which provides automatic differentiation capabilities.

`Pyomo` does not support expressions involving complex numbers, nor does it implement matrix-matrix or matrix-vector operations. Therefore, all constraint functions and the objective function have to be implemented using expressions of real variables, and matrix operations have to be implemented by looping over components. In appendix B, expressions for all used constraints can be found, involving only real numbers.

The lack of support for complex variables and matrix operations, makes the implementation slightly more involved. However, this is made up for by the automatic differentiation capabilities that come with `Pyomo`, eliminating the need for closed form formulas for derivatives of the objective and constraint function. Especially derivatives of constraint functions with respect to tap variables can have very complicated expressions.

7.3. Initial Guess

The algorithm described in Section 7.1 requires an initial guess $\mathbf{x}^{(0)}$ for the optimization variables. All OPF problems that we consider, are scenarios for which a power flow calculation can be performed, meaning that all the exogenous variables are given, and that an initial value for the controllable variables is provided (for exogenous and controllable variables, see Section 5.2.1).

With the fixed and controllable variables given, we can compute a power flow to obtain an initial guess for the state variables. This ensures that the initial guess satisfies the power flow equation equality constraints. It does not ensure that any other constraints are satisfied, however, and it therefore also does not ensure a feasible solution to the OPF problem exists.

Alternatively, the state variables (and possibly the controllable variables) can be initialized with a flat start, meaning that all power injections and voltage angles are set to 0, and that all voltage magnitudes are set to 1.

We expect that an OPF calculation that is started from a power flow takes fewer iterations to converge, than an OPF calculation that is started from a flat start, since a significant share of the constraints are already satisfied.

7.4. Discrete Variables

In Section 5.4, we mentioned that sometimes variables cannot always take a continuos range of values, but only a discrete set of values (e.g., only integer values). This is in particular the case for shunt step variables and transformer tap variables. This makes the problem significantly harder.

To obtain a solution for the discrete problem, we use the simple rounding method that is described in [34, sec. 2.1]. To be able to use our method in practical applications, it is necessary to obtain a discrete solution. However, to restrict the scope of this thesis, we did not consider any other methods. This means that this method will most probably not obtain globally optimal solutions for the discrete problem.

We assume that the problem is formulated in such a way that the discrete variables may only take integer values. For example, a transformer whose tap position is indicated by some integer between -5 and 5 (rather than the variable representing the ratio and taking values from e.g. 0.95 to 1.05).

First we obtain a solution for the continuous relaxation. Now every variable that is required to be discrete, is rounded to the nearest integer value, and is fixed to this value (i.e., it is now an exogenous variable, as described in Section 5.2.1). Then another round of optimization is performed, where the discrete variables are fixed, and the remaining variables are optimized once more. We use the solution of the first optimization round as the initial guess for the second round. Optionally, the dual variables can also be initialized with the resulting dual variables of the optimization round. This is supported by `Ipopt` and is called a *warm start*. When the second optimization run uses a warm start, it can be a good idea to start that round with a lower value of μ, since we expect the initial guess to be close to the optimal solution.

8

Convergence Problems

Although the methodology described in Chapter 7 performs well on some networks, it does not converge on others. In this chapter, we look at what problems arise when performing OPF calculations, and how they can be fixed.

8.1. What are Convergence Problems

The methodology presented in Chapter 7 could go wrong in a number of different ways. In this case, the exit message of `Ipopt` will signal where in the algorithm an error occurred. This error message can give a good indication of why no optimal solution was found. The error message could for example indicate that the problem is locally infeasible, or that it is unbounded. In that case, an investigation is needed if this is indeed the case, for example by manually trying to find a feasible point. The problem can then be changed in such a way that it is no longer infeasible or unbounded (e.g. by relaxing constraints, changing the objective or adding additional bounds to variables).

There are some situations in which an OPF problem is feasible, and has a bounded set of optimal solutions, but still fails to converge. It might for example be the case that we have an initial guess that satisfies all constraints, and that all variables are bounded below and above. We refer to behaviour, where we know an optimal solution exists, but our algorithm struggles to find it, as *convergence problems*. In this chapter, two causes of converge problems are treated.

8.2. High Admittance Edges

In Section 7.1.3, we have seen that an ill-conditioned iteration matrix can cause problems in the calculation of the Newton step. In this section we run through a practical example of a grid model that causes ill-conditioned iteration matrices.

A high condition number indicates that a matrix is almost singular[1]. Conversely, a matrix that is almost singular, for example because it has columns that are very nearly linearly dependent, has a high condition number. This mechanism is the reason that edges with an exceptionally high admittance, compared to other edges can cause difficulties. As we will illustrate, edges with a high admittance (i.e., edges where $|Y_{kl}^{\text{sr}}|$ is large relative to other edges) cause the columns of the matrix block A to be nearly linearly dependent.

We focus our attention on the first and second derivatives of the constraint function $\mathbf{c}$. The constraints that make up this function can vary, but in any case the nodal power equality constraints are included, that is, equations (4.14) and (4.15). We have repeated them here for convenience, and denote them by

[1] For the 2-norm this can be made precise, see e.g. [27, sec. 2.6.2].

g_k^P and g_k^Q respectively.

$$g_k^P(\mathbf{x}) := P_k - \sum_{l \in \mathcal{N}} |V_k||V_l| \left(G_{kl}\cos(\delta_k - \delta_l) + B_{kl}\sin(\delta_k - \delta_l)\right) = 0 \tag{8.1}$$

$$g_k^Q(\mathbf{x}) := Q_k - \sum_{l \in \mathcal{N}} |V_k||V_l| \left(G_{kl}\sin(\delta_k - \delta_l) - B_{kl}\cos(\delta_k - \delta_l)\right) = 0. \tag{8.2}$$

In a state that resembles a standard operating state, we expect the voltage magnitudes $|V_k|$ and $|V_l|$ to be around 1. Therefore, the magnitude of the derivatives of these functions are very much dependent on the magnitude of G_{kl} and B_{kl}, and therefore of the magnitude of the admittance matrix component $|Y_{kl}|$. Now suppose that for node k, there is one component Y_{ki} that has a much higher magnitude than any other component in row k, caused by a very high admittance of edge k, l. Then the functions g_k^Q and g_k^P will be dominated by the term that corresponds to column i. This, in turn, has the effect that the partial derivatives of g_k^Q and g_k^P with respect to $|V_k|, \delta_k, |V_i|$ and δ_i are much bigger than all other partial derivatives of g_k^Q and g_k^P. A possible effect of this is that the rows of A become very nearly linearly dependent. We illustrate this with two examples. The first example shows that high admittance edges can cause particular columns in the block A of our iteration matrix to become very nearly linearly dependent. The second example runs through two small scale OPF calculations, one with a high admittance edge, and one without. We show that the model with the high admittance edge indeed performs worse than the one without the high admittance edge.

8.2.1. Jacobian of Constraints

We first investigate the influence of a high admittance edge on the Jacobian matrix of the constraints. We show with a calculation that this small grid leads to two nearly dependent columns in the Jacobian. Note that we reason this in the context of optimal power flow, but that the reasoning also holds for power flow calculations, since the Jacobian matrix in (4.22) is very similar to A.

Consider the scenario that is shown in Figure 8.1: a grid consisting of 4 nodes, connected by 3 edges. One of the edges is a switch, with a very high admittance, the other two are lines with a much lower admittance. Suppose line 0 and line 1 have a purely imaginary admittance of $B_1 \cdot j$ and $B_3 \cdot j$

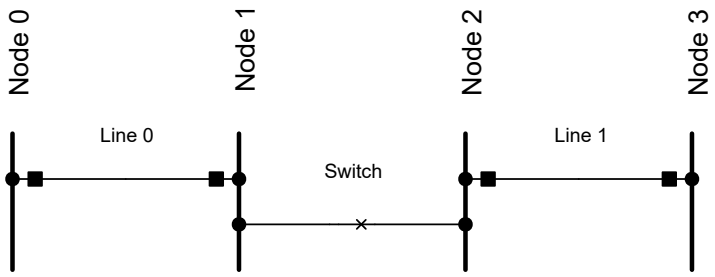

Figure 8.1: A grid consisting of 4 nodes and 3 edges. Nodes 1 and 2 are connected by a switch with a high admittance.

respectively, and the switch has a purely imaginary admittance of $B_2 \cdot j$. Furthermore, we assume that $|V_0| = \cdots = |V_3| = 1$ and $\delta_0 = \cdots = \delta_3 = 0$ (this corresponds to a flat start initial guess). Then the partial derivatives of g_1^P and g_2^P are as follows:

$$\frac{\partial g_1^P(\mathbf{x})}{\partial \delta_0} = -B_1, \qquad \frac{\partial g_1^P(\mathbf{x})}{\partial \delta_1} = B_1 + B_2, \qquad \frac{\partial g_1^P(\mathbf{x})}{\partial \delta_2} = -B_2,$$

$$\frac{\partial g_2^P(\mathbf{x})}{\partial \delta_1} = -B_2, \qquad \frac{\partial g_2^P(\mathbf{x})}{\partial \delta_2} = B_2 + B_3, \qquad \frac{\partial g_2^P(\mathbf{x})}{\partial \delta_3} = -B_3.$$

All other partial derivatives are 0. This leads to the following two columns in the Jacobian A, which contain the partial derivatives of g_1^P and g_2^P:

$$\begin{bmatrix} -B_1 & 0 \\ B_1 + B_2 & -B_2 \\ -B_2 & B_2 + B_3 \\ 0 & -B_3 \end{bmatrix}. \tag{8.3}$$

Here we have excluded all 0 rows. These two columns are not linearly dependent. However, if the admittance of the switch (B_2) is much bigger than the admittances of the lines (B_1 and B_3), then we could neglect B_1 and B_3, and make the following approximation:

$$\begin{bmatrix} -B_1 & 0 \\ B_1 + B_2 & -B_2 \\ -B_2 & B_2 + B_3 \\ 0 & -B_3 \end{bmatrix} \approx \begin{bmatrix} 0 & 0 \\ B_2 & -B_2 \\ -B_2 & B_2 \\ 0 & 0 \end{bmatrix}. \tag{8.4}$$

Note that the approximation has two linearly dependent columns. The matrix A, which contains the columns in the left matrix, therefore has two columns that are very close to being linearly dependent, and is therefore ill-conditioned, when B_2 is much bigger than B_1 and B_3. The same phenomenon occurs for the partial derivatives of g_1^Q and g_2^Q with respect to the voltage magnitudes.

8.2.2. Two Models of a Grid With a Switch

In Section 8.2.1 we have shown how the presence of a high admittance edge can cause the Jacobian of the constraints to become ill-conditioned. In this section, we extend the example to a more complete and slightly more realistic example of a grid, on which we can perform an OPF calculation. We study the effect of the high admittance edge on the iteration matrix.

We compare two models of the 5 node grid shown in figure 8.2. The grid consists of 5 nodes, connected by 4 edges. Two of the nodes have a load connected, and one has a PV generator attached. Two of the edges represent a high voltage line and one represents a transformer. The middle edge represents a switch, which has a very high admittance. We compare the model where we treat the grid as is, by considering it as a 5 node grid with 1 high admittance edge, versus the model where we merge the middle 2 nodes (Node 1 and Node 4 in the figure) and treat it as a 4 node grid. We show that our method performs worse on the 5 node model, than on the 4 node model.

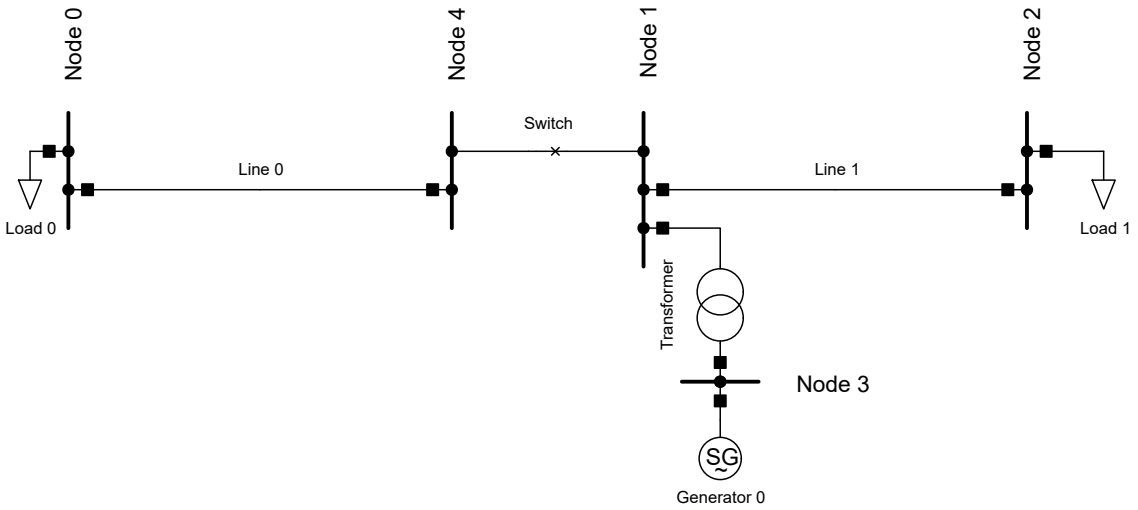

Figure 8.2: A grid consisting of 5 nodes and 4 edges. Nodes 1 and 4 are connected by a switch and can be merged into a single node.

Both lines have a susceptance of 1, the transformer has a susceptance of 50, and the switch has a susceptance of 10^{10}. The conductance of each edge is 10% of the reactance of the edge. Below are the admittance matrices of both models, in (8.5) of the 4 node model, and in (8.6) of the 5 node model.

$$(0.1 + 1j)\begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -52 & 1 & 50 \\ 0 & 1 & -1 & 0 \\ 0 & 50 & 0 & -50 \end{bmatrix} \tag{8.5}$$

$$(0.1 + 1j)\begin{bmatrix} -1 & 0 & 0 & 0 & 1 \\ 0 & (-51 - 10^{10}) & 1 & 50 & 10^{10} \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 50 & 0 & -50 & 0 \\ 0 & 10^{10} & 0 & 0 & -10^{10} \end{bmatrix}. \tag{8.6}$$

The admittance matrix in (8.6) clearly has two columns with very big outliers in terms of the magnitude of the components, caused by the high admittance of the switch.

8.2.3. The Iteration Matrix

We analyse the iteration matrix for both models. The iterate that is considered is the iterate after one iteration, with a flat start as initial guess. In Figure 8.3 the magnitudes of the elements of both iteration matrices can be seen. We can see that in the right iteration matrix, the range of magnitudes (approx. 10^{-3} to 10^{11}) is much larger than in the left iteration matrix (approx. 10^{-3} to 10^{4}). As we expect from the reasoning in Section 8.2.1, the particularly large values in the top right block of the iteration matrix of the 5 node grid (those in light green) are the partial derivatives of $g_1^{\mathrm{P}}, g_1^{\mathrm{Q}}, g_4^{\mathrm{P}}$ and g_4^{Q} with respect to $|V_1|, \delta_1, |V_4|$ and δ_4.

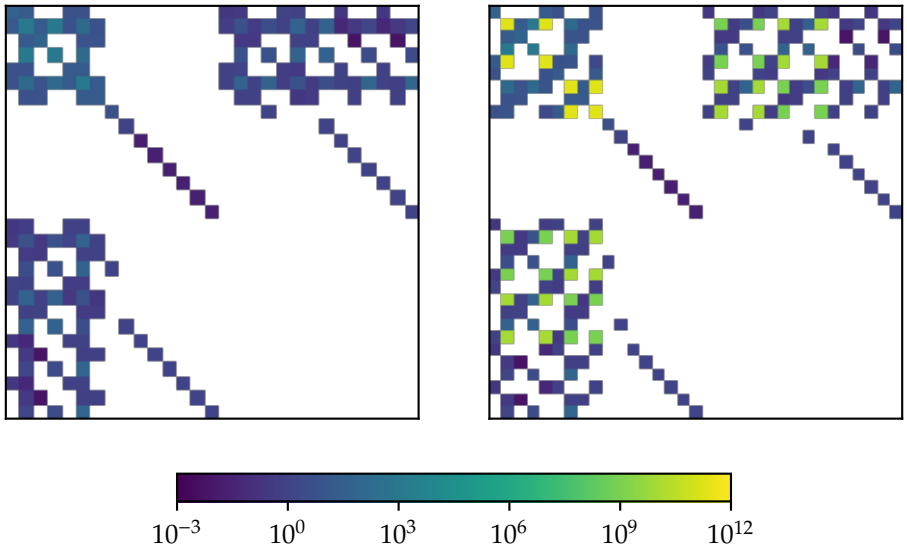

Figure 8.3: Magnitude of the components of the iteration matrices. On the left for the 4 node model, on the right for the 5 node model. White indicates a zero element.

Due to these very high values, some rows, and columns of the iteration matrix of the 5 node model are very nearly linearly dependent. This is reflected in a very high condition number: $3.9 \cdot 10^{13}$, compared to $2.7 \cdot 10^{5}$ for the iteration matrix of the 4 node model (the condition numbers are in the 2-norm). When we look at the top right block of both iteration matrices (i.e., block A), we can also see the difference. The

condition number of the block A is $8.34 \cdot 10^{10}$ for the 5 node model, compared to $4.78 \cdot 10^{2}$ for the 4 node model. This confirms that the high condition numbers are indeed caused by an ill-conditioned block A.

8.2.4. Effect on Ipopt Iterations

The fact that this matrix is so ill-conditioned leads to problems in the calculation of the optimal solution to this OPF problem. The method described in Chapter 7 converges within 9 iterations for the 4 node model, but fails to converge for the 5 node grid. At some point the algorithm encounters a very ill-conditioned matrix, and enters the restoration phase, which fails.

When working with models with high admittance edges, we observed that the behaviour of the method is very sensitive to changes in the input of the method. Small changes to the model (e.g., changes to the fixed variables, admittance of lines, etc), the initial guess, or to the solver settings (e.g., changes to initial value of μ, changing scaling options, changing the used linear solver), can lead to totally different outcomes. Such a small change could make the difference between `Ipopt` converging to an optimal solution, exiting with an error message (for example because of an unsuccessful restoration phase), or iterating seemingly indefinitely. When a model has high admittance edges, the behaviour of the method is very unpredictable and results are difficult to reproduce.

8.2.5. Cut-off for High Admittance

What exactly counts as a *high admittance* edge? At what magnitude does the admittance of an edge become problematic for convergence? There is no straightforward answer to this question, as it depends on factors such as the size of the grid, the relative magnitude of the admittances in the grid, and the ratio of the conductance and the susceptance of each edge.

To give a rough idea: we found that on a grid with around 1500 nodes and 2000 edges, our method performed bad when the range of magnitudes of the non-zero components of the admittance matrix was $1.5 \cdot 10^{-1}$ to $6 \cdot 10^{10}$. However, when we applied our method to the same model, modified such that the admittances were in the range $1.5 \cdot 10^{-1}$ to $1.5 \cdot 10^{6}$, it worked well, without convergence issues.

In practice, our approach is not to have a cut-off value above which an edge is considered to be a high-admittance edge, but rather to consider the physical component that the edge represents, to be an indicator of whether the edge is considered a high admittance edge. In our experiments, the methods proposed in Section 8.2.6 were applied to all edges representing a switch.

8.2.6. Dealing with High Admittance Edges

As already suggested by the comparison between the 4 and 5 node model, one way to remedy issues with high admittance edges, is to consider nodes that are connected by a high admittance edge as a single node. In this section, we introduce two more ways of dealing with high admittance edges. However, first, we consider the effect of scaling.

Effect of Scaling

The comparison in Section 8.2 was done without performing NLP scaling or iteration system scaling. When we enable the default NLP scaling that `Ipopt` performs, an optimal solution is found for the 5 node model in 9 iterations, the same amount as for the 4 node model. In some situations, scaling can therefore help the algorithm converge. However, it is not a universal solution to the issues caused by high admittance edges. The fact that it works in some situations and not in others, contributes to the unpredictability of the `Ipopt` algorithm on networks with these issues.

Merging Nodes

In Section 8.2 we saw that by merging nodes (i.e., considering them as a single node) that are connected by high admittance edges, we can get rid of the problematic edges. Merging of nodes is also commonly done for power flow calculations, as high admittance edges can also cause issues during power flow calculations. Procedures to obtain a grid model with merged nodes are readily available (see e.g., [50, sec. 3.j] and [18, sec. 6.2.1]).

This method has two potential downsides. The first, is that it changes the underlying model. Two nodes that can potentially have slightly different voltages, are suddenly represented by a single node, and therefore a single voltage. This will change the voltages and currents throughout the model, and

therefore, some error is introduced to the solution of an optimal power flow problem. In most situations, however, this error is so small that it can be neglected. The voltage of nodes that are connected by edges with a very high admittance are almost identical in normal operating conditions.

The second downside, is that there is no straightforward way to obtain the current flowing through the high admittance edge in question. It is therefore not possible to put a constraint on the current flowing through the edge, or to use the value in the objective function. If the amount of current flowing is only required as part of the result, then a power flow calculation could be performed on the original model with the OPF results (provided that the high admittances edges do not cause issues in the power flow calculation).

Decreasing Admittance
Another way to get rid of high admittances, is simply by decreasing their admittance. This could for example be done by scaling the admittance of all high admittance edges, so that the magnitudes of the admittances are below some specified maximum.

Just like the node merging method, the underlying model is changed, and therefore this method also has the downside that an error is introduced to the solution of the OPF. Whether the error is better or worse than when using the node merging method, depends on the specific problem and the specified maximum.

With this method, it is possible to put a constraint on the amount of current flowing through an edge, or to use this value in the objective.

8.3. Parallel Transformers

A second issue that can cause convergence problems, is the optimization of the tap positions of parallel transformers. When two transformers connect the same pair of nodes, and their tap ratios are optimized with two independent variables, convergence of our method can be significantly hindered.

During our research, this behaviour presented itself in some situations on larger grids. However, it was hard to reproduce on a smaller grid. Therefore, we study the behaviour on a particular grid, namely the model of the Dutch high voltage grid. For comparison, we also consider a modified version of the 118 Bus grid (see Section C.1). On the Dutch grid model, the parallel transformers caused very slow convergence. On the 118 Bus grid model, the convergence speed was also impacted, but not as significantly as on the Dutch grid.

Running an OPF calculation[2] on the Dutch grid, without optimizing parallel transformers (i.e., the tap variables of those transformers were fixed), the method converges in 150 iterations. When the parallel transformers are also included in the optimization, the number of iterations required increases to 692. This is an increase of more than 300%, even though the number of transformer tap variables increased by about 17% (129 tap variables vs. 151 tap variables), and the increase in the total number of optimization variables is less than 1% (2475 optimization variables vs. 2497 optimization variables). For the 118 Bus network, this is much less pronounced. The calculation requires 47 iterations without parallel transformers and 59 iterations with parallel transformers.

One major difference with the convergence problems caused by high admittance edges, is that the behaviour is much less unpredictable. In our experience, having separately controlled parallel transformers can cause very slow convergence, but besides that, the method does generally still converge.

In the next section, we have a closer look at how parallel transformers behave in AC power flow calculations.

8.3.1. Current Flow in Parallel Transformers

Suppose we have two identical parallel transformers, labelled 1 and 2, between two nodes labelled k and l. Suppose they both have a tap on the side of node k and the tap ratio of the transformers is denoted by t_1 for transformer 1 and t_2 for transformer 2. Both t_1 and t_2 are elements of $\mathbb{C}$. We denote the current

[2]The OPF calculation that was performed is very similar to the one in Chapter 9a. The only difference being that all transformers were controllable, instead of only those between transmission nodes.

flowing into transformer 1 and 2 by I_1^{f} and I_2^{f}, respectively. Then,

$$I_1^{\mathrm{f}} = V_k \frac{Y^{\mathrm{sr}} + \frac{Y^{\mathrm{es}}}{2}}{t_1^2} + V_l \frac{Y^{\mathrm{sr}}}{t_1} \tag{8.7}$$

$$I_2^{\mathrm{f}} = V_k \frac{Y^{\mathrm{sr}} + \frac{Y^{\mathrm{es}}}{2}}{t_2^2} + V_l \frac{Y^{\mathrm{sr}}}{t_2}. \tag{8.8}$$

We could consider the pair $(I_1^{\mathrm{f}}, I_2^{\mathrm{f}})$ as a vector in $\mathbb{C}^2$. The total current out of node k into both transformers is $I_1^{\mathrm{f}} + I_2^{\mathrm{f}}$, this is the inner product of the vector with $(1, 1)$. The component of the vector that is perpendicular to $(1, 1)$, (i.e., in the direction of $(1, -1)$), does not contribute anything to the current from node k towards node l. This means that a change of the vector $(I_1^{\mathrm{f}}, I_2^{\mathrm{f}})$ in the direction of $(1, -1)$ does not change anything about the state of the system, except for the value of I_1^{f} and I_2^{f}. A more physical interpretation of this is that the inner product of the vector with $(1, -1)$ is $I_1^{\mathrm{f}} - I_2^{\mathrm{f}}$, which can be interpreted as current flowing from transformer 1 into transformer 2, via node k.

In general, we want to avoid currents that are not contributing to the transmission of power. Unnecessary currents cause components to heat up, and be closer to their operating limits. Therefore, we would like to make sure that $I_1^{\mathrm{f}} - I_2^{\mathrm{f}} = 0$. For identical transformers, this is the case whenever $t_1 = t_2$. Operating identical parallel transformers in such a way that their tap ratio is the same, is therefore optimal, in the sense that no unnecessary currents are flowing through the transformers.

Now suppose we set $t_1 = t_2$. Then $I_1^{\mathrm{f}} = I_2^{\mathrm{f}}$, so

$$I_1^{\mathrm{f}} + I_2^{\mathrm{f}} = 2\left(V_k \frac{Y^{\mathrm{sr}} + \frac{Y^{\mathrm{es}}}{2}}{t_1^2} + V_l \frac{Y^{\mathrm{sr}}}{t_1}\right) \tag{8.9}$$

$$= V_k \frac{2Y^{\mathrm{sr}} + \frac{2Y^{\mathrm{es}}}{2}}{t_1^2} + V_l \frac{2Y^{\mathrm{sr}}}{t_1}. \tag{8.10}$$

This is equivalent to a single tapped transformer with twice the series admittance and twice the shunt admittance as the original transformers. Therefore, the behaviour of two parallel transformers with the same tap setting is the same as the behaviour of a single larger transformer.

8.3.2. Dealing with Parallel Transformers

We saw that we can dramatically speed up OPF calculations by excluding parallel transformers from the optimization. However, unless the tap variables happen to be fixed to their optimal value, this will lead to an optimal solution with a worse objective value. In Section 8.3.1 we have seen that when the tap position of identical parallel transformers are equal, no currents are flowing that are not contributing to the total power flow. Furthermore, the pair of transformers acts identically to a single transformer. This suggests that we could set the tap positions to be equal, without worsening the objective value of the optimal solution, and have a model that acts the same as a model without parallel transformers.

There are two obvious ways to ensure that the tap positions of parallel transformers are equal. The first is by adding an equality constraint that ensures the tap variables are equal. The second way, is by using a single tap variable per set of parallel transformers. The latter method has the benefit that it does not need any extra constraints, and it decreases the amount of optimization variables.

This gives us three ways of modifying the OPF problem, to reduce the amount of calculation time: not including parallel transformers, setting the parallel transformer taps equal with equality constraints, and optimizing parallel transformers with a single tap variable. In Table 8.1, the three methods are compared with the original OPF problem.

All three methods successfully decrease the number of iterations that is required. The first method, that excludes the parallel transformers from the problem seems to work slightly better in decreasing the number of iterations than the other two methods. For the 118 Bus model, the first, third, and fourth calculation (parallel transformers, equality constraint and single variable) converged to the same optimal solution, the OPF calculation with the parallel transformers excluded converged to a different solution,

calculation	Dutch Grid Model		118 Bus Model	
	# iterations	# tap variables	# iterations	# tap variables
parallel transformers	692	151	61	18
no parallel transformers	150	129	39	0
equality constraint	155	151	43	18
single variable	170	138	40	9

Table 8.1: Number of iterations for the original OPF calculations, compared to the three modified OPF calculations.

with a worse objective value. This means that the equality constraint method and the single variable method successfully decreased the number of iterations, without the objective value getting worse.

For the Dutch grid model, the first calculation had the best objective value, then the second calculation, and then the third and fourth, which converged to the same solution. The fact that the second calculation had a better result than the third and fourth is inconsistent with the reasoning that we can set the taps of parallel transformers equal. However, when we redo the third calculation, with the solution of the second calculation as an initial guess, it converges to a different solution, with an objective value that is better than the second calculation. This suggests that the third and fourth calculation are a local minimum of the OPF problem. The differences between all the obtained solutions was minor, however, with the total reactive power injection (which was the objective for these calculations) differing less than 1.5 Mvar between the best and worst solution. It should also be noted that not all parallel transformers are identical, with some pairs of transformers having slightly differing admittances.

9a

Results

In this chapter, we perform some calculations on an OPF problem that mimics a real life reactive power dispatch scenario. We first describe the problem and then show the results of the OPF calculation. Finally, we also perform OPF calculations after slightly modifying the OPF problem, to see the effect of those modifications on the results.

9a.1. Problem Description

The situation that we mimic in this section, is one that might occur in the operating room of a high voltage grid. For a specific hour of the day, a forecast has been made on the generation and the consumption of all parties that are connected to the grid. A few hours ahead, the operators have to decide on how they will operate the transformers and shunts, and they have to provide voltage set-points for all generators for that hour. They would like to operate the grid with minimal cost, while satisfying all requirements that ensure a safe and reliable operation of the grid.

9a.1.1. Grid Model

This simulation is done on a modified version of the *IEEE 118 bus* model [24], a model loosely based on the American power system in December 1962. The model has been modified slightly, the details of this can be found in Appendix C. The model consists of a total of 137 buses, split over 7 nominal voltage levels: the transmission levels 230 kV and 500 kV, and the generation levels 13.8 kV, 15.5 kV, 20 kV, 22 kV and 24 kV. The grid has a total of 198 edges, 170 of which are lines, and 28 of which are 2-winding transformers.

9a.1.2. Variables

The model has a total of 581 variables, 227 of which are fixed, so 360 optimization variables. Table 9a.1 gives an overview of the amount and type of each variable.

variable	total	fixed	controllable	state
nodal voltage magnitude	137	0	54	83
nodal voltage angle	137	1	0	136
active power injection generator	54	44	10	0
reactive power injection generator	54	0	0	54
active power injection load	91	91	0	0
reactive power injection load	91	91	0	0
shunt step	14	0	14	0
transformer tap	9	0	9	0

Table 9a.1: Amount of variables of each type in the considered OPF problem.

Nodal Voltages The voltage magnitude and voltage angle at each node are state variables, except at the nodes where a generator is attached, there they can also be controllable or fixed, depending on the generator that is attached (see the paragraph about generator injections and voltages). Bounds are only applied to the voltage magnitudes at the nodes at transmission level (i.e., those with nominal voltage 230 kV and 500 kV). The voltage magnitude at these nodes should be between 0.9 and 1.1 pu.

Generator Injections and Voltages The model contains 54 generators, 20 of these model a synchronous compensator, that can only inject reactive power.

The model includes a maximal active power injection for each generator[1] and a minimal active power injection of 0. We use these as upper and lower bound of the active power injection of each generator. To model a system with a distributed slack, we let the active power injection of the 10 generators[2] with the highest maximal active power injection (which includes the slack generator) be controllable, and fix the active power injection of all other generators.

All generators have controllable voltage magnitude. The reactive power injection of all generators is a state variable, and it is bounded above and below[3]. The voltage angle is fixed to 0 for the slack generator[4], and is a state variable that is not bounded for all other generators.

Load Injections The model contains 91 loads, all attached to a node with a nominal voltage of 230 kV. All loads have a fixed active and reactive power injection.

Shunt Steps The model contains 14 shunts, all of which are capacitive shunts, which means that they can only supply reactive power. They are all attached to nodes with a nominal voltage level of 230 kV. All of them are controllable, and their step variable can be between 0 and 1.

Transformer Taps The model contains 28 transformers, 9 of which have a tap shifter. All tap shifters are real tap shifters (i.e., $t_{kl} \in \mathbb{R}$), meaning that no phase shifting is done by the transformers. Only the transformers between nodes at transmission levels have a tap shifter. In practice, all other transformers are under control of customers connected to the transmission grid, not under control of the TSO themselves, so this is a realistic scenario. The minimum and maximum tap positions for all transformers are -1 and 1, respectively.

9a.1.3. Continuity of Variables

In this model, both the shunt step positions and transformer tap postions can only take integer values. However, we mostly consider the continuous relaxation (i.e., we assume that all variables are continuous). Only in Section 9a.2.3, we obtain a solution with the discrete variables at integer positions.

9a.1.4. Constraints

For each node, the nodal power equality constraints (as described in Section B.1.1) apply. For each edge, the full current constraints apply (as described in Section B.1.3).

9a.1.5. Objective

We would like to minimize the following objective:

$$f(\mathbf{x}) = \sum_{g \in \mathcal{G}} \left| Q_g \right|, \tag{9a.1}$$

where $\mathcal{G}$ is the set of all generators[5]. This models a scenario in which generators are compensated equally for every unit of reactive power they consume or supply. This objective is also described in [12,

[1] Attribute `e:Pmax_uc` in PowerFactory.
[2] Only considering the generators that are modelled as a synchronous machine in the PowerFactory model.
[3] Attributes `t:Q_min` and `t:Q_max` in PowerFactory
[4] This is Gen 69 in the PowerFactory model.
[5] Recall from Section 7.1.1 that $\mathbf{x}$ is the vector of all controllable and state variables. Therefore, all Q_g are components of $\mathbf{x}$.

p. 23]. However, the absolute value is not differentiable at 0, so we use $x \mapsto \sqrt{x^2 + \epsilon}$ as an approximation of the absolute value, where ϵ is a small positive number. This leads to the following objective:

$$f(\mathbf{x}) = \sum_{g \in \mathcal{G}} \sqrt{Q_g^2 + \epsilon}. \tag{9a.2}$$

The smaller the value of ϵ, the closer this function is to (9a.1). However, our method makes use of a second order approximation of the objective function (i.e., the Hessian of f is used to move towards a minimum of f), and this second order approximation is less accurate around 0 if ϵ is small. Therefore, a small ϵ can lead to slow convergence. We use the value[6] of $\epsilon = 10^{-2}$. In Section 9a.3.4, the behaviour of the OPF calculations is compared for a few different values of ϵ.

9a.1.6. Initial Guess

Before the OPF calculation, a power flow calculation is performed. The result of this calculation is used as the initial guess for the OPF calculation.

The initial guess satisfies the nodal power equality constraints, since it is the result of a power flow calculation. However, it is not a feasible point, since not all inequality constraints are satisfied. For 12 of the generators, the initial reactive power injection is not within the bounds specified for that variable, and 4 edges are loaded above the maximum, with a total of 8 current constraints that are violated (for all edges both the to- and from- constraint are violated).

9a.1.7. Test Setup

All calculations in this chapter are performed with `Ipopt` version 3.14.17 [21], with linear solver `MA27` [30]. The calculations are performed on a laptop with a 12 core Intel i5-1350P processor and 16 GB of ram.

9a.2. Results

In this section, we present the results of an OPF calculation on the problem described in Section 9a.1. Both the numerical cost and the resulting solution of the calculation are presented.

9a.2.1. Numerical Cost

Our method successfully obtains a solution to the OPF problem after 29 iterations. According to the timing reported by `Ipopt`, this takes 0.031 seconds, 0.011 of which are spent evaluating the objective and constraint functions, and its derivatives. If we also include the time it takes for `Pyomo` to send the problem to `Ipopt`, and loading the result back into the model, solving the problem takes 1.16 seconds[7]. There is some overhead setting up the problem as well, with the construction of the `Pyomo` model taking around 0.17 seconds, and the calculation of the initial guess taking around 0.25 seconds[8].

9a.2.2. Variables and Objective

We compare the initial guess (before OPF) with the obtained solution (after OPF).

The objective value after optimization is 10.80, compared to 32.72 before optimization. This corresponds to a reduction from 3139 Mvar to 855 Mvar of total reactive power injection. The net reactive power injection (i.e., reactive power supplied minus reactive power consumed) is also reduced, from 575 Mvar to 339 Mvar. In Figure 9a.1, a histogram of the reactive power injections before and after OPF can be seen.

[6] Recall from Section 3.1.4 that the unit for Q is 100 Mvar. So when a generator injects 1 Mvar, its contribution to the objective is $\sqrt{0.01 + \epsilon} = \sqrt{0.02}$.

[7] This was measured by executing the solver via `Pyomo` in a Jupyter notebook cell with the `timeit` command.

[8] This also includes the overhead of moving back and forth between nodal injections and load and generator injections, which is described in Section 5.2.1.

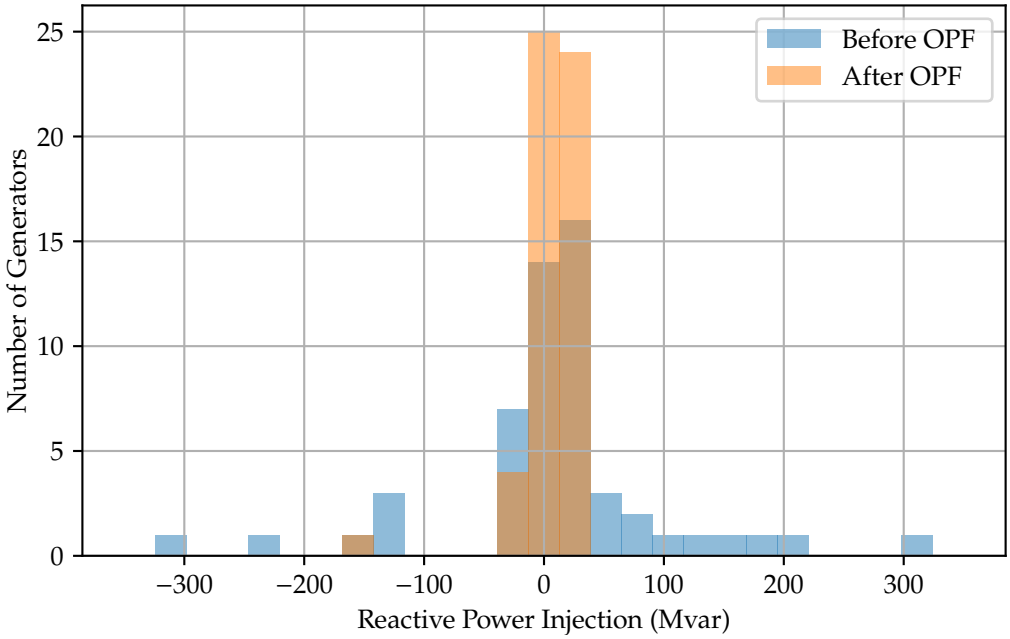

Figure 9a.1: Histogram of the reactive power injections of the generators. Values before and after the OPF calculation are shown.

The amount of reactive power injected by the capacitive shunts in the network is also reduced, even though the objective function does not directly incentivize the reduction of reactive power injection by shunts. The transformer tap ratios all increase after OPF, most of them to the max position of 1. In Figure 9a.2 the shunt and transformer variables can be seen before and after OPF.

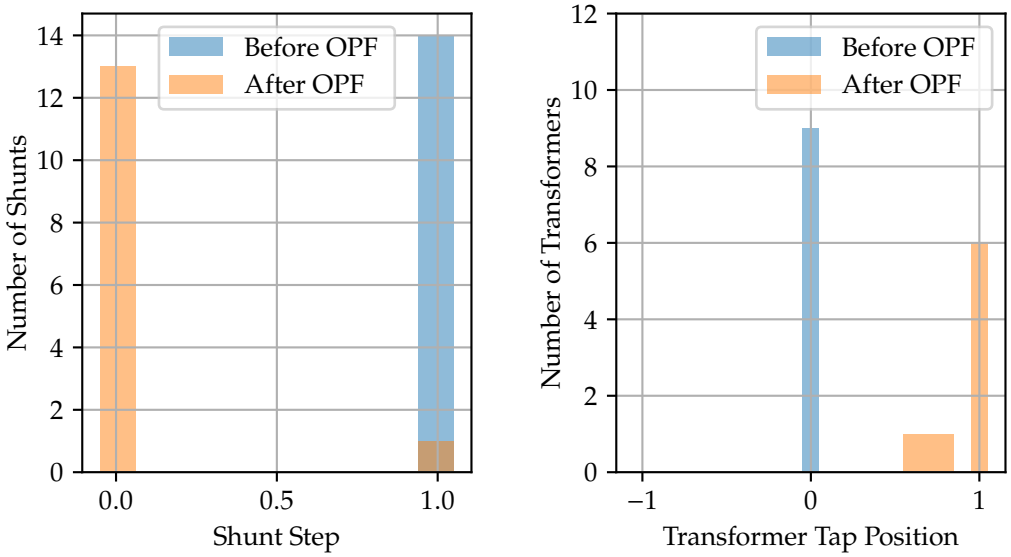

Figure 9a.2: Histogram of shunt step position and transformer tap position before and after OPF.

A histogram of the voltage magnitudes at all transmission nodes can be seen in Figure 9a.3. Globally, the nodal voltage magnitudes seem to become a bit higher after the OPF calculations. This seems to disagree with the intuition that a reduced net reactive power supply leads to decreasing voltages. Locally, the voltages seem to follow the intuition, with voltage magnitudes rising in the neighbourhood

of generators with an increased reactive power injection, and vice versa. The global phenomenon could be due to the increase in lines with a low load. The reactive power losses in a line depend on the amount of current through a line. With a low amount of current flowing, a line will supply reactive power, but above a certain threshold will start consuming reactive power instead [44, p. 13].

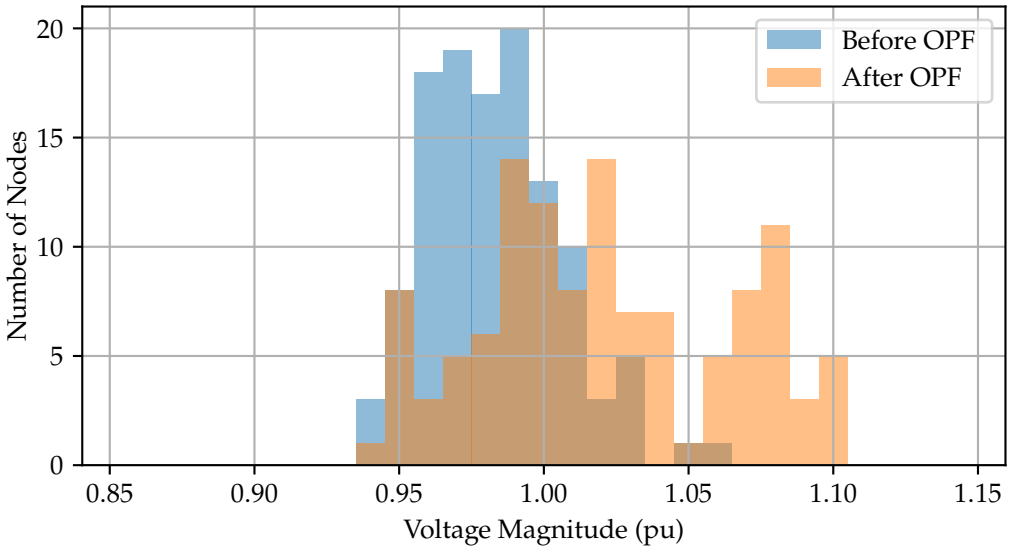

Figure 9a.3: Histogram of the voltage magnitude of transmission nodes before and after OPF.

In Figure 9a.4, a histogram of the loading of all edges is shown. The loading of edge $\{k, l\}$ is calculated as follows:

$$\frac{\max\{|I_{kl}^{\mathrm{f}}|, |I_{kl}^{\mathrm{t}}|\}}{I_{kl}^{\max}} \cdot 100\%, \tag{9a.3}$$

so that both the from- and to-constraint are satisfied when the loading is below[9] 100%.

[9] In the result of an OPF calculation it can occur that an edge is loaded slightly above 100%. This can happen when the edge loading constraint is binding (i.e., it is at its upperbound). The edge loading is then indeed above 100%, but it is below 100% + ϵ, where ϵ is the tolerance for `Ipopt` up to which all constraints need to be satisfied. In figure 9a.4, the bin of 90% - 100% actually runs until 100 + 10^{-6}%, so that binding constraints fall within this bin.

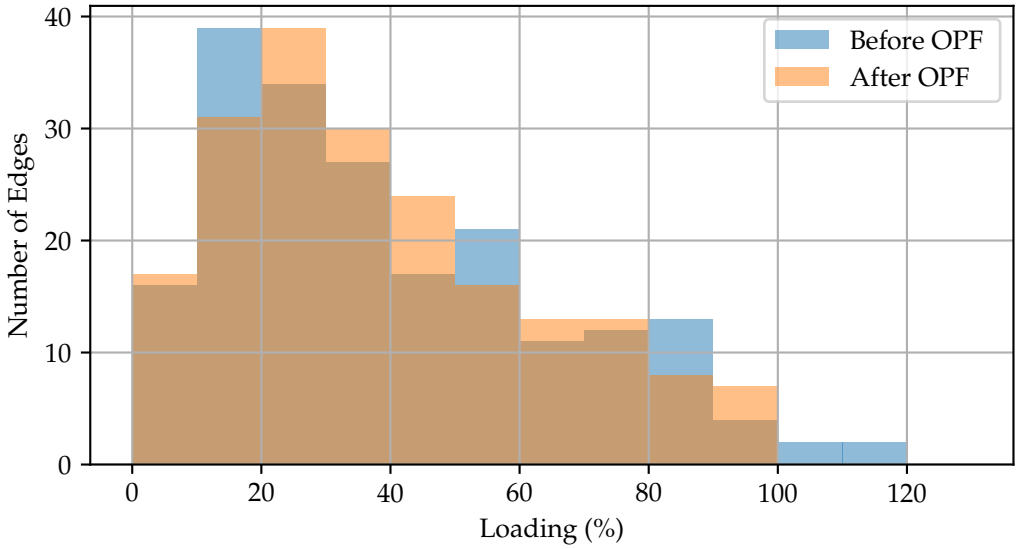

Figure 9a.4: Histogram of the edge loading before and after OPF.

9a.2.3. Simple Rounding Method

We now perform the simple rounding method, as described in Section 7.4. As we can see in Figure 9a.2, most of the discrete variables are either at their upper bound or lower bound in the result of the optimization. This means that only 3 variables need to be rounded off, namely 3 tap position variables.

The second optimization round, with the discrete variables fixed, takes another 27 iterations, and 1.15 seconds. When we warm start the second optimization run with the initial value for μ set to 10^{-6}, we can bring this down to 8 iterations. However, this still takes about 1.13 seconds.

The effect on the objective is minimal, with the objective value now being 10.81, compared to the 10.80 of the solution of the continuous relaxation. The total reactive power injection is 855.18 Mvar, compared to 1.5460 before rounding.

The solutions of the continuous relaxation and the solution after the simple rounding method are very close. The biggest difference in reactive power injection by a generator is 0.7 Mvar, and the biggest difference in nodal voltage magnitudes is 0.018 pu.

9a.3. Comparison Between Scenarios

In this section, we perform some variations of the OPF calculation of Section 9a.2. We change the variables, constraints, and objective. The goal is to show the flexibility of the method, allowing it to be applied in different scenarios. If, for example, not all components can be controlled, then this can be incorporated into the model with a simple change. At the same time, this section indicates what effects such changes might have on the performance of the method and the obtained solution.

In total, we compare 16 variations of the OPF calculation, labeled 1 to 16. Included in this are the initial guess (i.e., before OPF), the original OPF calculation, which we described in Section 9a.1, and the original OPF calculation with simple rounding, which we described in Section 9a.1. In Table 9a.2 we describe how each variation differs from the original scenario. Table 9a.3 shows the computational cost of each variation, as well as some numbers that characterise the solution of that variation.

The convergence time shown in Table 9a.3 is the average time of 7 runs[11]. The standard deviation of these 7 runs was mostly between 0.02 and 0.06 seconds each time. Except for scenario 10., the convergence times are therefore very close to each other, and the differences between them are not very significant. It is therefore more reliable to look at the iteration count of each scenario if we want to compare performance.

[11] As measured by the `timeit` command in a Jupyter notebook.

Variation	Description
1.	Before OPF
2.	Original scenario
3.	Original scenario with simple rounding method
4.	A flat start is used as initial guess.
5.	We fix all shunt step variables to their initial values
6.	We fix all transformers tap variables to their initial values
7.	We fix all shunt step variables and all transformer tap variables to their initial values
8.	We fix the reactive power injection of 33 out of 53 generators[10]. The voltage magnitude of these generators becomes a state variable.
9.	We change the bounds for nodal voltage magnitude to 0.92 and 1.08.
10.	We change the bounds for nodal voltage magnitude to 0.95 and 1.05
11.	We reduce the current capacity for each edge to 90% of its original value
12.	We reduce the current capacity for each edge to 87% of its original value
13.	We set ϵ in the objective function to 10^{-4}
14.	We set ϵ in the objective function to 10^{-6}

Table 9a.2: Description of all variations. The modifications are made to the original scenario, as described in Section 9a.1.

scenario	# optimization variables	# iterations	convergence time (s)	objective value	$\sum_g \lvert Q_g \rvert$ (Mvar)
1.	-	-	-	32.72	3139
2.	360	29	1.16	10.80	855
3.	360	29 + 8	1.19 + 1.13	10.81	855
4.	360	38	1.17	10.80	855
5.	346	29	1.22	12.44	1033
6.	351	41	1.19	11.15	887
7.	337	57	1.17	12.79	1069
8.	327	29	1.18	21.42	1970
9.	360	23	1.17	10.89	864
10.	360	33	1.17	11.10	885
11.	360	41	1.19	11.28	885
12.	360	28	1.16	16.60	1428
13.	360	64	1.22	8.21	807
14.	360	1101	3.54	8.03	802

Table 9a.3: Results of all variations.

9a.3.1. Initial Guess

In the original scenario, the result of a power flow calculation was used as an initial guess for the OPF calculation. Variation 4, used a flat start initial guess (see Section 7.3) instead. In Section 7.3, we explained that expect the calculation that starts from a flat start to take longer than one that starts from the result of a power flow calculation. Both OPF calculations converge to the same solution, and when we compare variation 4 with variation 2, we see that, indeed, the flat start takes longer to converge to the solution.

9a.3.2. Effect of fixing variables

In variations 5 to 8 we exclude the shunts, transformers and a part of the generators from the optimization by fixing their variables. We compare these variations with the original scenario (variation 2). Firstly, we notice that a decrease in the number of variables can lead to an increase or decrease in the number of iterations. When we fix the transformers (variation 5 and 6), the number of variables decreases, but the number of iterations increases. When we only fix the shunts (variation 4), or we fix a subset of the reactive power injections (variation 7), the number of iterations does not change. Whether each iteration takes longer with more variables, is not possible to say from this data.

Fixing variables always makes the objective value larger. Keeping the transformers fixed (variation 6) has a lower impact on the objective value than keeping the shunts fixed (variation 5). This could be explained by the fact that there are more shunts than transformers (14 shunt step variables vs. 9 transformer tap variables). However, the effects of fixing either or both the shunts and the transformers on the objective value are small if we compare it to fixing a part of the reactive power injections of the generators, which has a big impact on the objective compared to the other variations.

9a.3.3. Effect of Tighter Constraints

In variations 9 to 12 we made the constraints tighter. We also performed an OPF calculation with the current capacity for each edge set to 85%, however, no feasible solution was found.

We see that tighter constraints lead to more iterations in some cases and to fewer iterations in other cases. The objective value gets worse as the constraints get tighter. Especially in variation 12, the objective is a lot worse.

Additionally, we see that tightening the constraints has a big effect on voltages. In Figure 9a.5 the voltage magnitudes at all transmission nodes of variation 11 are compared with those of the original scenario. The voltages are shifted closer towards the upper bound at 1.1 pu. This makes sense, since the same power can be transferred with a lower current if the voltage is higher.

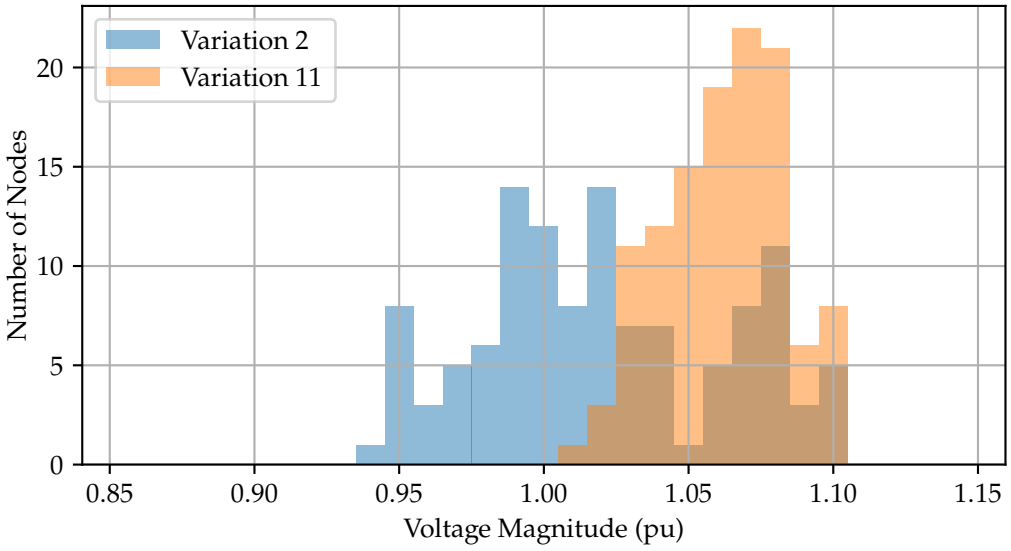

Figure 9a.5: Histogram of voltage magnitude of transmission nodes after an OPF calculation for varaiations 2 and 11.

9a.3.4. Effect of ϵ in Objective

In the original scenario, the objective function is the following:

$$\sum_{g \in \mathcal{G}} \sqrt{Q_g^2 + \epsilon}. \tag{9a.4}$$

In Section 9a.1.5, we explained that this is a surrogate for the actual objective, which is the following:

$$\sum_{g \in \mathcal{G}} |Q_g|. \tag{9a.5}$$

When comparing the initial guess (variation 1) with the result of the OPF calculation (variation 2), we can see that by minimizing (9a.4), the actual objective (9a.5) is also greatly reduced. This was done with a value of $\epsilon = 10^{-2}$. When we decrease the value of ϵ, the surrogate objective becomes a better approximation for the actual objective, and this reflects in the results. Comparing scenario 2 with scenarios 12 and 13, we see that the value of (9a.5) becomes smaller, as ϵ gets smaller. When we set $\epsilon = 10^{-6}$, the reactive power injected by generators is reduced by an extra 6%, compared to when $\epsilon = 10^{-2}$. However, we can also see that the number of iterations required to obtain this result greatly increases. This aligns with our expectation, which we described in Section 9a.1.5.

9a.3.5. Minimizing Losses

In this section, we compare two more variations in which we change the objective. In the first we minimize the grid losses, by minimizing the active power injection of all the generators for which the active power injection is not fixed. The objective is as follows:

$$\sum_{\{g \in \mathcal{G} : P_g \text{ is not fixed}\}} P_g. \tag{9a.6}$$

In the second we minimize the following objective:

$$\sum_{i \in \mathcal{N}} (|V_i| - 1)^2. \tag{9a.7}$$

With this objective, we try to get the voltage magnitude at all nodes as close to 1 as possible.

We compare the variations with the before OPF variation, and the original calculations. Table 9a.4 gives an overview of the variations. The results can be seen in Table 9a.5.

Variation	Description
1.	Before OPF
2.	Minimize Reactive power injection (original scenario)
15.	Minimize grid losses
16.	Minimize deviation of voltages from 1 pu.

Table 9a.4: Description of the variations to the objective.

scenario	# optimization variables	# iterations	convergence time (s)	$\sum_g \lvert Q_g \rvert$ (Mvar)	$\sum_g P_g$ (MW)	$\sum_i (V_i - 1)^2$ (pu)
1.	-	-	-	3139	3802	0.112
2.	360	29	1.16	855	3792	0.335
15.	360	26	1.14	2108	3772	0.975
16.	360	26	1.13	2766	3793	0.018

Table 9a.5: Results of the variations to the objective.

The reactive power injection is significantly higher in variations 15 and 16, than in variation 2. It is still lower than in variation 1, however. When we minimize the active power losses, the losses can be reduced by 30 MW compared to variation 1 and 20 MW compared to variation 2.

The change in objectives has a significant influence on the voltage magnitudes throughout the grid, as we can see in Figure 9a.6. When we minimize the deviation from 1, the voltages are, of course, concentrated around 1 pu. When we minimize the losses, the voltages are close to their upper bound of 1.1 pu. This is the same behaviour as we saw in Section 9a.3.3 when the capacity of the edges was reduced. If the reactive power injection is minimized, the voltages are more spread out than in the other variations.

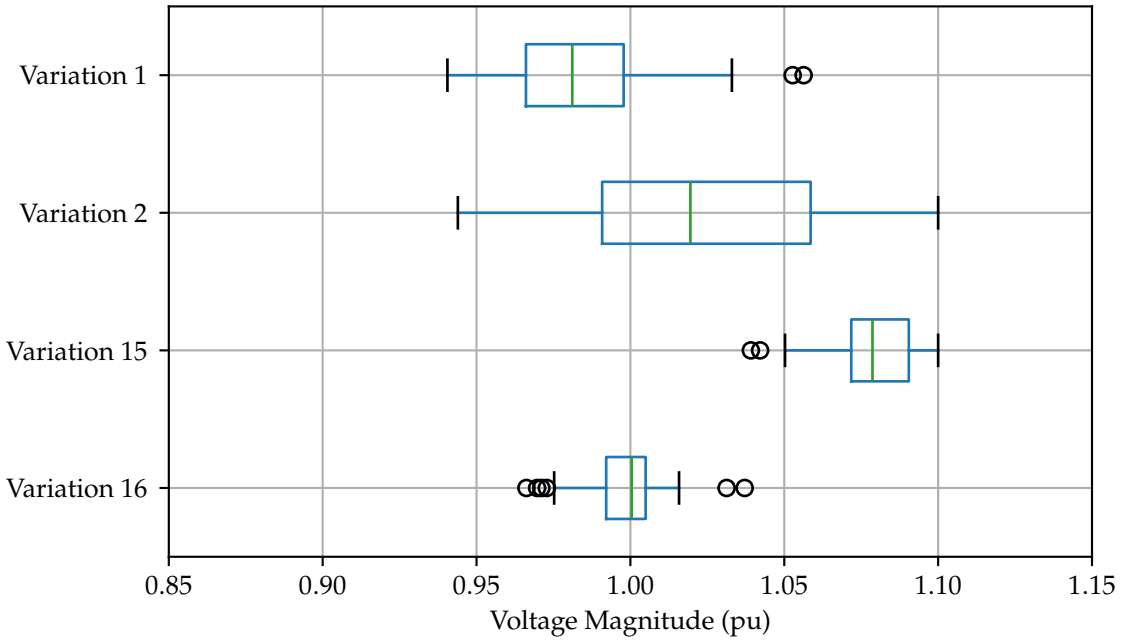

Figure 9a.6: Boxplot of the voltage magnitude of transmission nodes in variation 1, 2, 15 and 16.

In terms of computational cost, the three calculations are very similar.

9a.4. Conclusion

In Sections 9a.1 and 9a.2, we have walked through an example of an OPF calculation, that simulates a reactive power dispatch scenario. Our method finds a solution in a few seconds. The solution indeed reduces the reactive power injection by generators, and all constraints are satisfied.

In Section 9a.3, we made several modifications to our original OPF problem, and studied the effect of these modifications on the performance of our method and the effects on the results. This provides some insights into the behaviour of the method and the model, and shows the flexibilty of the implemented method. General statements on how to optimally control this grid cannot be made from these comparisons.

9b

Results on the Dutch Grid

This chapter is parallel to Chapter 9a. It shows the same calculations, but on a model of the Dutch high voltage grid instead. The reason that not all calculations were done on the Dutch grid in the first place, is that the model used in this chapter is a confidential model, and some results obtained by calculations on this model are not suitable for publication. Any results that reveal or imply how the network is operated, or how it could be operated, may be redacted. Results on the model's computational cost are not redacted, whenever possible.

9b.1. Problem Description

The constraints and objective are similar to those in section 9a. We describe the details.

9b.1.1. Grid Model

This simulation is done on the `2024_04_11 INTRADAY` model. The calculations have been performed on operation scenario `2024_04_11/6_12`. Synchronous generator [redacted] was selected as the reference machine.

The model consists of a total of 1132 nodes. Of these, 816 nodes are at the transmission level, with nominal voltages 110 kV, 150 kV, 220 kV and 380 kV. The grid has a total of 1502 edges.

9b.1.2. Variables

The model has a total of 4554 variables, 2070 of which are fixed, so 2484 optimization variables. Table 9b.1 gives an overview of the amount and type of each variable.

variable	total	fixed	controllable	state
nodal voltage magnitude	1132	0	81	1051
nodal voltage angle	1132	1	0	1131
active power injection generator	109	99	10	0
reactive power injection generator	109	0	28	81
active power injection load	985	985	0	0
reactive power injection load	985	985	0	0
shunt step	93	0	93	0
transformer tap	8	0	9	0

Table 9b.1: Amount of variables of each type in the considered OPF problem.

Nodal Voltages The voltage magnitude and voltage angle at each node are state variables, except at the nodes where a generator is attached, there they can also be controllable or fixed, depending on the generator that is attached (see the paragraph about generator injections and voltages). Bounds are only applied to the voltage magnitudes at the nodes at transmission level (i.e., those with nominal voltage

110 kV, 150 kV, 220 kV and 380 kV). The bounds on the voltage magnitude at each level are given in Table 9b.2.

nominal voltage level	lower bound (pu)	upper bound (pu)
110 kV	0.9	1.1
150 kV	0.9	1.1
220 kV	0.95	1.1
380 kV	0.95	1.1
other	-	-

Table 9b.2: Bounds on the voltage magnitude for each nominal voltage level.

Generator Injections and Voltages The model contains 109 generators. The model includes a minimal and maximal active power injection for each generator[1]. We use these as upper and lower bound. To model a system with a distributed slack, we let the active power injection of the 10 generators with the highest nominal power[2] (which includes the slack generator) be controllable, and fix the active power injection of all other generators.

All generators have either a controllable voltage magnitude, or a controllable reactive power injection, dependent on whether they are a PV or a PQ machine. When the voltage magnitude is controllable, the reactive power injection is a state variable, and vice versa. The reactive power injection is bounded above and below[3]. The voltage angle is fixed to 0 for the slack generator[4], and is a state variable that is not bounded for all other generators.

Load Injections The model contains 985 loads. All loads have a fixed active and reactive power injection.

Shunt Steps The model contains 93 shunts, 37 of which are capacitive shunts, and 56 are reactive shunts. All of them are controllable, and their step variable can be between 0 and 1.

Transformer Taps The model contains 205 (2-winding) transformers. Only the transformers between nodes at transmission levels are controllable, all other transformers are fixed. In total 10 transformers are controllable. Two of these are in parallel to each other, and they are controlled with a single tap variable. This means that we have a total of 9 tap variables. Of the 10 controllable transformers, 5 have a purely real tap changer, the other 5 have a complex tap changer. The transformers have between 21 and 32 possible tap positions.

9b.1.3. Constraints

For each node, the nodal power equality constraints (as described in Section B.1.1) apply. For each edge, the full current constraints apply (as described in Section B.1.3).

9b.1.4. Objective

Just like in Chapter 9a, the objective function is as follows:

$$f(\mathbf{x}) = \sum_{g \in \mathcal{G}} \sqrt{Q_g^2 + \epsilon}, \tag{9b.1}$$

with $\epsilon = 10^{-2}$.

[1] Attributes `e:Pmin_uc` and `e:Pmax_uc` in PowerFactory.
[2] Attribute `Pmax_a` in PowerFactory.
[3] Attributes `t:Q_min` and `t:Q_max` in PowerFactory
[4] This is [redacted] in the PowerFactory model.

9b.1.5. Initial Guess

Before the OPF calculation, a power flow calculation is performed. The result of this calculation is used as the initial guess for the OPF calculation.

[redacted]

9b.1.6. Test Setup

The test setup is the same as in Chapter 9a.

9b.2. Results

In this section, we present the results of an OPF calculation on the problem described in Section 9b.1. Both the numerical cost and the resulting solution of the calculation are presented.

9b.2.1. Numerical Cost

Our method successfully obtains a solution to the OPF problem after 34 iterations. According to the timing reported by `Ipopt`, this takes 0.469 seconds, 0.264 of which are spent evaluating the objective and constraint functions, and its derivatives. If we also include the time it takes for `Pyomo` to send the problem to `Ipopt`, and loading the result back into the model, solving the problem takes 2.21 seconds. There is some overhead setting up the problem as well, with the construction of the `Pyomo` model taking around 1.62 seconds, and the calculation of the initial guess taking around 2.8 seconds[5]..

9b.2.2. Variables and Objective

We compare the initial guess (before OPF) with the obtained solution (after OPF).

The objective value after optimization is [redacted] 9b.1, [redacted]

[redacted] All shunt steps are either at their lower bound 0, or at their upper bound 1 after OPF. [redacted] For the transformers, 5 tap positions are either at their upper or lower bound, and 4 are not. The increase or decrease in the shunt step variables and the transformer tap positions can be seen in Figure 9b.2

A histogram of the voltage magnitudes at all transmission nodes can be seen in Figure 9b.3. [redacted]

In Figure 9b.4, a histogram of the loading of all edges is shown. [redacted]

9b.2.3. Simple Rounding Method

We now perform the simple rounding method, as described in Section 7.4. As mentioned in Section 9b.2.2, only 4 transformer tap variables need to be rounded off. All other discrete variables are at their bounds, which is an integer position.

The second optimization round, with the discrete variables fixed, takes another 37 iterations, and 2.07 seconds. When we warm start the second optimization run with the initial value for μ set to 10^{-6}, we can bring this down to 14 iterations and 1.81 seconds.

[5] This also includes the overhead of moving back and forth between nodal injections and load and generator injections, which is described in Section 5.2.1.

Figure 9b.1: Histogram of the reactive power injections of the generators. Values before and after the OPF calculation are shown.

The effect on the objective is minimal, with the objective value now being [redacted] 9a.1.5. [redacted]

9b.3. Comparison Between Scenarios

The convergence time shown in Table 9b.4 is the average time of 7 runs. The standard deviation of these 7 runs was mostly between 0.02 and 0.1 seconds each time. The convergence times are therefore very close to each other, and the differences between them are not very significant. It is therefore more reliable to look at the iteration count of each scenario if we want to compare performance.

9b.3.1. Initial Guess

In the original scenario, the result of a power flow calculation was used as an initial guess for the OPF calculation. In Section 7.3, we explained that the benefit of this, is that the intial guess already satisfies the equality constraints. In variation 4, we used a flat start initial guess (see Section 7.3) instead, however, this time the OPF calculation not converge. After 223 iterations the algorithm switches to emergency mode, which fails. This indicates once again that our method performs better with an initial guess that is the result of a power flow calculation.

9b.3.2. Effect of fixing variables

In variations 5 to 8 we exclude the shunts transformers and a part of the generators from the optimization by fixing their variables. We compare these variations with the original scenario (variation 2). Similar to Chapter to the 118 Bus grid, a decrease in the number of variables can lead to an increase or decrease in the number of iterations.

[redacted]

Variation	Description
1.	Before OPF
2.	Original scenario
3.	Original scenario with simple rounding method
4.	A flat start is used as initial guess.
5.	We fix all shunt step variables to their initial values
6.	We fix all transformers tap variables to their initial values
7.	We fix all shunt step variables and all transformer tap variables to their initial values
8.	We fix the reactive power injection of 45 out of 109 generators[6]. The voltage magnitude of these generators becomes a state variable.
9.	We change the bounds for nodal voltage magnitude to[7] 0.92 and 1.08.
10.	We change the bounds for nodal voltage magnitude to 0.95 and 1.05
11.	We reduce the maximum current for each edge to 90% of its original value
12.	We reduce the maximum current for each edge to 85% of its original value
13.	We set ϵ in the objective function to 10^{-4}
14.	We set ϵ in the objective function to 10^{-6}
15.	We change the objective to $\sum_{i \in \mathcal{N}} (V_i - 1)^2$

Table 9b.3: Description of all variations. The modifications are made to the original scenario, as described in Section 9b.1.

scenario	# optimization variables	# iterations	convergence time (s)	objective value	$\sum_g \lvert Q_g \rvert$ (Mvar)	$\sum_i (V_i - 1)^2$ (pu)
1.	-	-	-	[redacted]	[redacted]	[redacted]
2.	2484	34	2.21	[redacted]	[redacted]	[redacted]
3.	2484	34 + 8	2.21 + 1.81	[redacted]	[redacted]	[redacted]
4.	2484	-	-	-	-	-
5.	2391	85	2.99	[redacted]	[redacted]	[redacted]
6.	2475	28	2.18	[redacted]	[redacted]	[redacted]
7.	2382	67	2.65	[redacted]	[redacted]	[redacted]
8.	2439	37	2.19	[redacted]	[redacted]	[redacted]
9.	2484	42	2.25	[redacted]	[redacted]	[redacted]
10.	2484	45	2.46	[redacted]	[redacted]	[redacted]
11.	2484	50	2.41	[redacted]	[redacted]	[redacted]
12.	2484	38	2.14	[redacted]	[redacted]	[redacted]
13.	2484	84	2.78	[redacted]	[redacted]	[redacted]
14.	2484	257	5.90	[redacted]	[redacted]	[redacted]
15.	2484	29	2.02	[redacted]	[redacted]	[redacted]

Table 9b.4: Results of all variations. The OPF calculation of variation 4 did not converge.

Figure 9b.2: Histogram of shunt step position and transformer tap position differences before and after OPF. A positive difference means an increase after OPF.

[redacted]

9b.3.3. Effect of Tighter Constraints

In variations 9 to 12 we made the constraints tighter. We see that tighter constraints lead to more iterations in some cases and to fewer iterations in other cases. The objective value gets worse as the constraints get tighter. The tighter constraints do not have a very big effect on the objective. On the 118 Bus network, this effect was more significant.

9b.3.4. Effect of ϵ in Objective

In the original scenario, the objective function is the following:

$$\sum_{g \in \mathcal{G}} \sqrt{Q_g^2 + \epsilon}. \tag{9b.2}$$

In Section 9a.1.5, we explained that this is a surrogate for the actual objective, which is the following:

$$\sum_{g \in \mathcal{G}} \left| Q_g \right|. \tag{9b.3}$$

When comparing the initial guess (variation 1) with the result of the OPF calculation (variation 2), we can see that by minimizing (9b.2), the actual objective (9b.3) is also greatly reduced. This was done with a value of $\epsilon = 10^{-2}$. When we decrease the value of ϵ, the surrogate objective becomes a better approximation for the actual objective, and this reflects in the results. Comparing scenario 2 with scenarios 13 and 14, we see that the value of (9b.3) is smaller, when we set $\epsilon = 10^{-4}$, however when we set $\epsilon = 10^{-6}$, it becomes slightly larger again. This could be due to a local minimum, because when we warm start the calculation of variation 14 with the solution of variation 13, it converges to a solution with objective [redacted] and the value of (9b.3) is [redacted].

Once again, the number of iterations and the execution time increases as ϵ gets smaller.

9b.3.5. Three Different Objectives

In this section, we compare two more variations in which we change the objective. In the first we minimize the grid losses, by minimizing the active power injection of all the generators for which the

Figure 9b.3: Histogram of the voltage magnitude of transmission nodes before and after OPF.

active power injection is not fixed. The objective is as follows:

$$\sum_{\{g \in \mathcal{G} : P_g \text{ is not fixed}\}} P_g. \tag{9b.4}$$

In the second we minimize the following objective:

$$\sum_{i \in \mathcal{N}} (|V_i| - 1)^2 . \tag{9b.5}$$

With this objective, we try to get the voltage magnitude at all nodes as close to 1 as possible.

We compare the variations with the before OPF variation, and the original calculations. Table 9b.5 gives an overview of the variations. The results can be seen in Table 9b.6.

Variation	Description
1.	Before OPF
2.	Minimize Reactive power injection (original scenario)
15.	Minimize grid losses
16.	Minimize deviation of voltages from 1 pu.

Table 9b.5: Description of the variations to the objective.

scenario	# optimization variables	# iterations	convergence time (s)	$\sum_g \lvert Q_g \rvert$ (Mvar)	$\sum_g P_g$ (MW)	$\sum_i (V_i - 1)^2$ (pu)
1.	-	-	-	[redacted]	[redacted]	[redacted]
2.	2484	34	2.21	[redacted]	[redacted]	[redacted]
15.	2484	43	2.24	[redacted]	[redacted]	[redacted]
16.	2484	29	2.02	[redacted]	[redacted]	[redacted]

Table 9b.6: Results of the variations to the objective.

[redacted]9a[redacted]

Figure 9b.4: Histogram of the edge loading before and after OPF.

In Figure 9a.6 the distribution of the voltages for all 4 variations can be seen.

Figure 9b.5: Boxplot of the voltage magnitude of transmission nodes in variation 1, 2, 15 and 16.

Just like for the 118 Bus grid, the three calculations are very similar in terms of computational cost.

9b.4. Conclusion

The results in this Chapter are quite similar to those of Chapter 9a, however there are some differences. The model of the Dutch grid seems to be slightly more challenging for OPF calculations. For example, in this chapter, we observed the method converging to a local optimum, and the method did not converge when starting from a flat start. Both of these behaviours where not observed during calculations on the Dutch grid. Other behaviour, such as the behaviour of the model under the different variations was very similar to the behaviour on the 118 Bus grid, however. Due to the slightly larger size of the model, the calculations take longer, but they still finish within a reasonable time.

This chapter shows that our method is able to do OPF calculations on a realistic model of the Dutch grid. The optimization can determine (locally) optimal set-points for components, for various objectives, and with various modifications to the variables and constraints.

10

Conclusion and Discussion

10.1. Conclusion

The aim of this thesis project is to implement an OPF algorithm that can perform OPF calculations on the Dutch grid. Quite some literature has been written on algorithms for Optimal Power Flow, and a handful of proprietary and open-source software packages exist to perform such calculations. Nonetheless, performing OPF calculations on models of real-world grids can still be difficult, because of convergence problems. That is, the method converges very slowly, or fails to converge to a solution, even though a solution exists.

In the implementation of an OPF algorithm for the Dutch high voltage grid, two causes of convergence problems were identified. Both originate from particular situations that can occur in the grid. The first is caused by components in the grid with a relatively high admittance (e.g., switches and breakers), that are modelled as an edge. The second cause that was identified, was trying to separately optimize the tap position of parallel tap transformers. Both of these convergence problems have been described in Chapter 8 and for both problems, multiple solution strategies have been found.

The result is a robust OPF method, that is able to perform OPF calculations on a real life model of the Dutch grid. This is an important step towards the application of OPF in operation of the Dutch transmission grid. We can optimize the control of all basic assets, such as generators, loads, transformers, and shunts. In Chapters 9a and 9b, the results of OPF calculations have been studied, along with the performance of the method for different variations of OPF problems.

10.2. Expanding the OPF Method

Before this model can be used in operation, there are a few features that are not yet supported. Firstly, we have not included a model for some of the more complex types of equipment that are being built into the grid. Some examples of this are 3-winding transformers with tap control, high-voltage direct current connections, synchronous compensators, FACTS devices, and secondary voltage or frequency controllers. Furthermore, the current models might also need to be expanded for certain situations. In unit commitment optimization, for example, generators can have a certain startup cost, which leads to a non-continuous cost function. This is not supported with our current model.

Transmission networks are required to operate in an N-1 secure state. This cannot be ensured with the current method. For this, the OPF model would have to be extended to a security constrained OPF model. This would greatly increase the practical usability of OPF calculations. However, as we explained in Section 5.3, this will come with new challenges, such as increased computational costs.

In the implementation of each of these expansions, new convergence problems might present themselves. After each addition to the model, it is important to carefully test the new capabilities, identify convergence problems, and find solutions. For an OPF method to have any applicability in real life operation of the grid, the method must be robust and reliable.

10.3. Topics for Additional Research

On the OPF problems that were tested, our method managed to obtain solutions that were at least locally optimal. However, there is reason to assume that these local optimal solutions are not always globally optimal. It is known that for certain types of OPF problems local optima exist, and in Section 8.3.2 we observed that an OPF calculation converged to a different solution, when the initial guess was changed. This suggests that, indeed, a local minimum was obtained. In this particular case the objective values of the obtained solutions were relatively close together, but we cannot rule out that there is a global minimum with a much better objective value. For a specific OPF problem this could be done by trying out different initial guesses and comparing the solutions that are obtained. Saying something about globally optimal solutions for general OPF problems would be much harder, however.

On the area of discrete OPF problems, the question of optimality is even more present. We implemented a method that was able to produce feasible solutions for the discrete problems on which we performed our experiments. However, this method cannot provide any guarantee about optimality. Furthermore, there might be discrete OPF problems for which a feasible solution exists, but where our method will not be able to obtain a solution.

The topic of sensitivity analysis could also offer interesting insights into the grids that are studied with OPF. The sensitivity of the objective function to the bounds of binding constraints, indicates which bounds would be the most beneficial to relax. This could, for example, indicate which components should be upgraded first to obtain a better objective value.

10.4. Applications of OPF

In Chapter 9a we restricted our attention to a single application of OPF, namely a reactive power dispatch scenario. Some aspects of this calculation could be made more realistic. For example, the objective function assumed an equal cost for the reactive power injection of all controllable generators. In practice, this might vary per generator, depending on the contracts that are in place between the TSO and the party operating the generator. Furthermore, the active power losses were not included in the objective function, even though this is also a cost that comes with the operation of the network. Improving the model by constructing more realistic objective functions could greatly improve the results that can be obtained with this method.

Optimal power flow is a broad class of problems that can also have many other applications, such as minimizing redispatch cost, or optimizing the placement of new assets. For these applications, an objective needs to be constructed, that accurately represents the real-life objective, constraints have to be determined, and a choice has to be made on which variables are controllable, and which should remain fixed.

References

[1] Kevin-Martin Aigner et al. "Solving AC Optimal Power Flow with Discrete Decisions to Global Optimality". In: *INFORMS Journal on Computing* 35.2 (2023). Cited by: 4, pp. 458–474. DOI: 10.1287/ijoc.2023.1270.

[2] O. Alsac et al. "Further Developments in LP-Based Optimal Power Flow". In: *IEEE Transactions on Power Systems* 5.3 (1990), pp. 697–711. DOI: 10.1109/59.65896.

[3] P.R. Amestoy et al. "A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling". In: *SIAM Journal on Matrix Analysis and Applications* 23.1 (2001), pp. 15–41.

[4] P.R. Amestoy et al. "Performance and Scalability of the Block Low-Rank Multifrontal Factorization on Multicore Architectures". In: *ACM Transactions on Mathematical Software* 45 (1 2019), 2:1–2:26.

[5] TenneT TSO B.V. *Grid Map Onshore Netherlands EN*. 2024. URL: https://www.tennet.eu/grid/grid-maps (visited on 12/12/2024).

[6] Daniel Bienstock et al. "Mathematical Programming Formulations for the Alternating Current Optimal Power Flow Problem". In: *4OR* 18 (Sept. 2020), pp. 249–292. DOI: 10.1007/s10288-020-00455-w.

[7] Stephen P Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge university press, 2004.

[8] Waqquas A Bukhsh et al. "Local solutions of the optimal power flow problem". In: *IEEE Transactions on Power Systems* 28.4 (2013), pp. 4780–4788.

[9] Michael L. Bynum et al. *Pyomo–optimization modeling in python*. Third. Vol. 67. Springer Science & Business Media, 2021.

[10] Richard H Byrd, Mary E Hribar, and Jorge Nocedal. "An interior point algorithm for large-scale nonlinear programming". In: *SIAM Journal on Optimization* 9.4 (1999), pp. 877–900.

[11] Richard H Byrd, Jorge Nocedal, and Richard A Waltz. *Knitro: An Integrated Package for Nonlinear Optimization*. 2005.

[12] Mary B Cain, Richard P O'neill, Anya Castillo, et al. "History of Optimal Power Flow and Formulations". In: *Federal Energy Regulatory Commission* 1 (2012), pp. 1–36.

[13] Florin Capitanescu. "Critical Review of Recent Advances and Further Developments Needed in AC Optimal Power Flow". In: *Electric Power Systems Research* 136 (July 2016), pp. 57–68. ISSN: 03787796. DOI: 10.1016/j.epsr.2016.02.008.

[14] Florin Capitanescu et al. "Contingency Filtering Techniques for Preventive Security-Constrained Optimal Power Flow". In: *IEEE Transactions on Power Systems* 22 (4 Nov. 2007), pp. 1690–1697. ISSN: 08858950. DOI: 10.1109/TPWRS.2007.907528.

[15] Florin Capitanescu et al. "Interior-point based algorithms for the solution of optimal power flow problems". In: *Electric Power Systems Research* 77 (5-6 Apr. 2007), pp. 508–517. ISSN: 03787796. DOI: 10.1016/j.epsr.2006.05.003.

[16] Florin Capitanescu et al. "State-of-the-Art, Challenges, and Future Trends in Security Constrained Optimal Power Flow". In: *Electric Power Systems Research* 81 (8 2011), pp. 1731–1741. ISSN: 03787796. DOI: 10.1016/j.epsr.2011.04.003.

[17] Jacques Carpentier. "Contribution a l'Etude du Dispatching Economique". In: *Bull. Soc. Fr. Elec. Ser.* 3 (1962), p. 431.

[18] Shravan Chipli. "Automating AC Power Flow Simulations". MA thesis. TU Delft, Aug. 2021.

[19] Carleton Coffrin et al. "PowerModels.jl: An Open-Source Framework for Exploring Power Flow Formulations". In: *2018 Power Systems Computation Conference (PSCC)*. June 2018, pp. 1–8. DOI: 10.23919/PSCC.2018.8442948.

[20] COIN-OR. *Basic Open-source Nonlinear Mixed INteger programming*. URL: https://www.coin-or.org/Bonmin/ (visited on 12/18/2024).

[21] COIN-OR. *Ipopt*. Version 3.14.17. Dec. 2024. URL: https://github.com/coin-or/Ipopt.

[22] Frank E. Curtis et al. *A Decomposition Algorithm for Large-Scale Security-Constrained AC Optimal Power Flow*. 2021. URL: https://arxiv.org/abs/2110.01737.

[23] Elnaz Davoodi and Florin Capitanescu. "A Robust Penalty-Based Approach to Optimal Reactive Power Dispatch With Discrete Controls". In: *2023 IEEE Belgrade PowerTech*. 2023, pp. 1–7. DOI: 10.1109/PowerTech55446.2023.10202758.

[24] Panayiotis Demetriou et al. "Dynamic IEEE Test Systems for Transient Analysis". In: *IEEE Systems Journal* 11 (4 July 2015), pp. 2108–2117. ISSN: 1932-8184. DOI: 10.1109/jsyst.2015.2444893.

[25] David Gay. *AMPL Solver Library (ASL)*. 2023.

[26] DigSILENT GmbH. *DIgSILENT PowerFactory 2022, User Manual*. Aug. 2022.

[27] Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.

[28] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2024.

[29] William E Hart, Jean-Paul Watson, and David L Woodruff. "Pyomo: modeling and solving mathematical programs in Python". In: *Mathematical Programming Computation* 3.3 (2011), pp. 219–260.

[30] HSL. *A collection of Fortran codes for large scale scientific computation*. URL: http://www.hsl.rl.ac.uk/.

[31] Karim Karoui et al. "Large Scale Security Constrained Optimal Power Flow". In: *2008 Power Systems Computation Conference (PSCC)*. July 2008.

[32] Karim Karoui et al. "New Large-Scale Security Constrained Optimal Power Flow Program Using a New Interior Point Algorithm". In: *2008 5th International Conference on the European Electricity Market*. 2008, pp. 1–6. DOI: 10.1109/EEM.2008.4579069.

[33] James L Kirtley. *Electric Power Principles: Sources, Conversion, Distribution, and Use*. Wiley, 2010. ISBN: 9780470667170.

[34] Peijie Li, Jianming Su, and Xiaoqing Bai. "An Objective Feasibility Pump Method for Optimal Power Flow with Unit Commitment Variables". In: *Electric Power Systems Research* 236 (Nov. 2024). ISSN: 03787796. DOI: 10.1016/j.epsr.2024.110928.

[35] Yuan Li et al. "Decision Making Under Uncertainty in Power System Using Benders Decomposition". PhD thesis. Iowa State University, 2008.

[36] Richard Lincoln. *PYPOWER*. 2023.

[37] Aaron Meurer et al. "SymPy: symbolic computing in Python". In: *PeerJ Computer Science* 3 (Jan. 2017), e103. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.103.

[38] Enrique Lobato Miguélez, Francisco M. Echavarren Cerezo, and Luis Rouco Rodrguez. "On the Assignment of Voltage Control Ancillary Service of Generators in Spain". In: *IEEE Transactions on Power Systems* 22 (1 Feb. 2007), pp. 367–375. ISSN: 08858950. DOI: 10.1109/TPWRS.2006.888984.

[39] A Monticelli and M V F Pereira. "Security-Constrained Optimal Power Flow with Post-Contingency Corrective Rescheduling". In: *IEEE Transactions on Power Systems* (1 1987).

[40] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

[41] Peter van Oirsouw and JFG Cobben. *Netten voor distributie van elektriciteit*. Phase to Phase, 2011.

[42] Ludovic Platbrood et al. "A Generic Approach for Solving Nonlinear-Discrete Security-Constrained Optimal Power Flow Problems in Large-Scale Systems". In: *IEEE Transactions on Power Systems* 29 (3 2013), pp. 1194–1203.

[43] KIOS Research and University of Cyprus Innovation Center of Excellence. *IEEE 118-bus modified test system*. URL: https://www.kios.ucy.ac.cy/testsystems/index.php/ieee-118-bus-modified-test-system/.

[44] Savu Crivat. Savulescu. *Real-time stability in power systems : techniques for early detection of the risk of blackout*. Uitleg over alles wat met voltage problems te maken heeft. Springer, 2006, p. 319. ISBN: 0387256261.

[45] Pieter Schavemaker and Lou van der Sluis. *Electrical Power System Essentials*. John Wiley and Sons, Incorporated, 2017. ISBN: 9781118803455.

[46] Baljinnyam Sereeter, Cornelis Vuik, and Cees Witteveen. "On a Comparison of Newton–Raphson Solvers for Power Flow Problems". In: *Journal of Computational and Applied Mathematics* 360 (Nov. 2019), pp. 157–169. ISSN: 0377-0427. DOI: 10.1016/J.CAM.2019.04.007.

[47] Brian Stott, Ongun Alsac, and Alcir J Monticelli. *Security Analysis and Optimization*. Dec. 1987.

[48] Brian Stott and Ongun Alsaç. "Optimal Power Flow - Basic Requirements for Real-Life Problems and their Solutions". In: 2012.

[49] The pandas development team. *pandas-dev/pandas: Pandas*. Version 2.14.10. Feb. 2020. DOI: 10.5281/zenodo.3509134.

[50] L. Thurner et al. "pandapower — An Open-Source Python Tool for Convenient Modeling, Analysis, and Optimization of Electric Power Systems". In: *IEEE Transactions on Power Systems* 33.6 (Nov. 2018), pp. 6510–6521. ISSN: 0885-8950. DOI: 10.1109/TPWRS.2018.2829021.

[51] Andreas Wächter and Lorenz T. Biegler. "On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming". In: *Mathematical Programming* 106 (1 May 2006), pp. 25–57. ISSN: 00255610. DOI: 10.1007/s10107-004-0559-y.

[52] Hongye Wang et al. "On Computational Issues of Market-Based Optimal Power Flow". In: *IEEE Transactions on Power Systems* 22 (3 Aug. 2007), pp. 1185–1193. ISSN: 08858950. DOI: 10.1109/TPWRS.2007.901301.

[53] Allen J Wood, Bruce F Wollenberg, and Gerald B Sheblé. *Power Generation, Operation, and Control*. Third edition. Wiley-IEEE, 2013. ISBN: 9781118733912.

A

Admittance Matrix for General Edge

In this chapter, we derive the admittance matrix for a general edge $\{k, l\} \in \mathcal{E}$ that is given in (3.17).

All transmission lines in the grid, whether overhead lines or underground cables, are modelled via the *standard pi-model* for a transmission line. For the details of this model, we refer to [45, app. E]. We model transformers as a pi-model line with an ideal transformer in series at one or both ends, as is shown in Figure A.1. If we set the transformer taps $t_{kl} = t_{lk} = 1$, we obtain back the standard-pi model. Therefore, the model in Figure A.1 can model both lines and transformers and is thus our model for a general edge. Although we do not use the exact same model, information on modelling transformers can be found in [45, app. B]. Since some transformers have a tap on both the low voltage and high voltage side, the model includes two ideal transformers.

Note that when we are working in a per unit normalized system, the ratio is also normalized. If we have, for example, a transformer between the buses k and l that are at their nominal voltages 110 kV and 220 kV, respectively, then the non-normalized tap ratio is $t_{kl} = 1/2$, but the normalized tap ratio is $t_{kl} = 1$.

Figure A.1 contains all variables in (3.17). Additionally, currents $I_1, \ldots, I_5$ are introduced for use in the derivation in this chapter (even though they share the notation, they are not related to current injections at nodes). Furthermore, we have also introduced two temporary nodes, k' and l', for the derivation. The voltages across these nodes are denoted by voltage phasors $V_{k'}$ and $V_{l'}$, respectively.

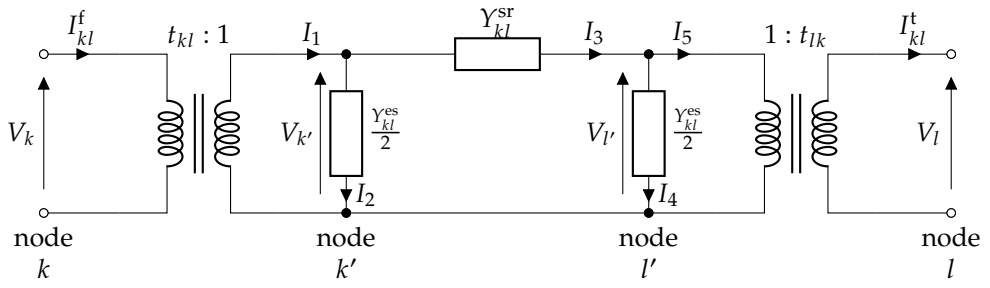

Figure A.1: Model for general edge, with two ideal transformers in series with a standard pi-model line. Admittances are shown for each load.

We note that voltages and currents on both sides of the transformers are related via the transformer tap

ratios t_{kl} and t_{lk}:

$$V_k = t_{kl} V_{k'} \tag{A.1}$$
$$V_l = t_{lk} V_{l'} \tag{A.2}$$
$$I_{kl}^{\mathrm{f}} = \frac{I_1}{{t_{kl}}^*} \tag{A.3}$$
$$I_{kl}^{\mathrm{t}} = \frac{I_5}{{t_{lk}}^*}. \tag{A.4}$$

Now, KCL gives us the following:

$$I_1 = I_2 + I_3 \tag{A.5}$$
$$I_5 = I_3 - I_4. \tag{A.6}$$

For the shunt capacitance of the line, we use Ohm's law:

$$I_2 = V_{k'} \frac{Y_{kl}^{\mathrm{es}}}{2} = \frac{V_k}{t_{kl}} \frac{Y_{kl}^{\mathrm{es}}}{2} \tag{A.7}$$
$$I_4 = V_{l'} \frac{Y_{kl}^{\mathrm{es}}}{2} = \frac{V_l}{t_{lk}} \frac{Y_{kl}^{\mathrm{es}}}{2}. \tag{A.8}$$

And similarly for the current I_3:

$$I_3 = Y_{kl}^{\mathrm{sr}}(V_{k'} - V_{l'}) = Y_{kl}^{\mathrm{sr}} \left(\frac{V_k}{t_{kl}} - \frac{V_l}{t_{lk}} \right). \tag{A.9}$$

Now if we combine (A.3), (A.5), (A.7) and (A.9), we get:

$$I_{kl}^{\mathrm{f}} = \frac{1}{{t_{kl}}^*} \left(\frac{V_k}{t_{kl}} \frac{Y_{kl}^{\mathrm{es}}}{2} + Y_{kl}^{\mathrm{sr}} \left(\frac{V_k}{t_{kl}} - \frac{V_l}{t_{lk}} \right) \right) \tag{A.10}$$
$$= \frac{1}{|t_{kl}|^2} \left(Y_{kl}^{\mathrm{sr}} + \frac{Y_{kl}^{\mathrm{es}}}{2} \right) V_k - \frac{Y_{kl}^{\mathrm{sr}}}{{t_{kl}}^* t_{lk}} V_l. \tag{A.11}$$

Similarly, we combine (A.4), (A.6), (A.8) and (A.9), to obtain:

$$I_{kl}^{\mathrm{t}} = \frac{1}{{t_{lk}}^*} \left(Y_{kl}^{\mathrm{sr}} \left(\frac{V_k}{t_{kl}} - \frac{V_l}{t_{lk}} \right) - \frac{V_l}{t_{lk}} \frac{Y_{kl}^{\mathrm{es}}}{2} \right) \tag{A.12}$$
$$= -\frac{1}{|t_{lk}|^2} \left(Y_{kl}^{\mathrm{sr}} + \frac{Y_{kl}^{\mathrm{es}}}{2} \right) V_l + \frac{Y_{kl}^{\mathrm{sr}}}{t_{kl} {t_{lk}}^*} V_k. \tag{A.13}$$

We can write this as the following matrix vector product:

$$\begin{bmatrix} I_{kl}^{\mathrm{f}} \\ -I_{kl}^{\mathrm{t}} \end{bmatrix} = \begin{bmatrix} \frac{1}{|t_{kl}|^2} \left(Y_{kl}^{\mathrm{sr}} + \frac{Y_{kl}^{\mathrm{es}}}{2} \right) & -\frac{Y_{kl}^{\mathrm{sr}}}{{t_{kl}}^* t_{lk}} \\ -\frac{Y_{kl}^{\mathrm{sr}}}{t_{kl} {t_{lk}}^*} & \frac{1}{|t_{lk}|^2} \left(Y_{kl}^{\mathrm{sr}} + \frac{Y_{kl}^{\mathrm{es}}}{2} \right) \end{bmatrix} \begin{bmatrix} V_k \\ V_l \end{bmatrix}. \tag{A.14}$$

This is exactly (3.17).

B

Constraint Expressions

This chapter includes expressions for all used constraint functions. The expressions all involve only real variables. All expressions involving voltages are expressed in terms of voltage magnitude $|V|$ and voltage angle δ, all expressions involving power, are expressed in terms of active power P and reactive power Q, and all expressions involving complex admittances are expressed in terms of conductance G and susceptance B.

For the derivation of the expressions in this chapter, we have made use of the symbolic mathematics library `SymPy` [37].

B.1. Constraints

B.1.1. Nodal Power Constraints

The nodal power equality constraints are the power flow equations that are derived in Section 4.2.2. We have repeated them here for completeness.

$$P_k = \sum_{l \in \mathcal{N}} |V_k||V_l|(G_{kl}\cos(\delta_k - \delta_l) + B_{kl}\sin(\delta_k - \delta_l)) \tag{B.1}$$

$$Q_k = \sum_{l \in \mathcal{N}} |V_k||V_l|(G_{kl}\sin(\delta_k - \delta_l) - B_{kl}\cos(\delta_k - \delta_l)). \tag{B.2}$$

Here P_k and Q_k are the nodal power injections. These may be substituted by the sum of the injections of the loads and generators attached to each node. This is described in Section 5.2.1.

B.1.2. Simplified Current Constraint

Looking at Figure A.1, there are multiple ways we could bound the current flowing through an edge. This constraint bounds the current I_3 in Figure A.1. A bound on this current has a relatively simple expression, and because of the symmetry of the edge model, only one bound is required to bound the current flowing in either direction. We therefore call it the *simplified current constraint*. The downside is that the current I_3 does not have a direct physical meaning, and the current flowing in or out of the edge (i.e. I_{kl}^{f} or I_{kl}^{t} in Figure A.1) could be much higher if the edge shunt admittance is high.

We use expression (A.9) and bound $|I_3|$ by $I_{kl}^{\max}$:

$$\left| Y_{kl}^{\mathrm{sr}} \left(\frac{V_k}{t_{kl}} - \frac{V_l}{t_{lk}} \right) \right| \leq I_{kl}^{\max} \tag{B.3}$$

We square both sides of this equation and express it in terms of real variables:

$$\left(B_{kl}^2 + G_{kl}^2\right)\left(|V_k|^2 - 2|V_k||V_l|\cos\left(\delta_k - \delta_l\right) + |V_l|^2\right) \leq (I_{kl}^{\max})^2 \tag{B.4}$$

B.1.3. Full Current Constraint

To obtain a more realistic bound on the current, it is better to bound the currents I_{kl}^{f} and I_{kl}^{t} in Figure A.1. The expressions for these constraints are a bit more complicated, and we need two constraints per edge[1] (one for I_{kl}^{f} and one for I_{kl}^{t}). We call these constraints the *full current constraints*.

First, we denote the four components in the matrix in (A.14) by $Y_{kl}^{\mathrm{ff}}, Y_{kl}^{\mathrm{ft}}, Y_{kl}^{\mathrm{tf}}, Y_{kl}^{\mathrm{tt}}$:

$$\begin{bmatrix} Y_{kl}^{\mathrm{ff}} & Y_{kl}^{\mathrm{ft}} \\ Y_{kl}^{\mathrm{tf}} & Y_{kl}^{\mathrm{tt}} \end{bmatrix} = \begin{bmatrix} \frac{1}{|t_{kl}|^2}\left(Y_{kl}^{\mathrm{sr}} + \frac{Y_{kl}^{\mathrm{es}}}{2}\right) & -\frac{Y_{kl}^{\mathrm{sr}}}{t_{kl}{}^{*} t_{lk}} \\ -\frac{Y_{kl}^{\mathrm{sr}}}{t_{kl} t_{lk}{}^{*}} & \frac{1}{|t_{lk}|^2}\left(Y_{kl}^{\mathrm{sr}} + \frac{Y_{kl}^{\mathrm{es}}}{2}\right) \end{bmatrix}. \tag{B.5}$$

Then, (A.11) becomes $I_{kl}^{\mathrm{f}} = Y_{kl}^{\mathrm{ff}} V_k + Y_{kl}^{\mathrm{ft}} V_l$. Now we apply the following bound:

$$|I_{kl}^{\mathrm{f}}| = |Y_{kl}^{\mathrm{ff}} V_k + Y_{kl}^{\mathrm{ft}} V_l| \leq I_{kl}^{\max}. \tag{B.6}$$

We square both sides and write it completely in real and imaginary parts:

$$\begin{aligned} &\left((B_{kl}^{\mathrm{ff}})^2 + (G_{kl}^{\mathrm{ff}})^2\right)|V_k|^2 + \left((B_{kl}^{\mathrm{ft}})^2 + (G_{kl}^{\mathrm{ft}})^2\right)|V_l|^2 \\ &\quad + 2|V_k||V_l|\left(\left(B_{kl}^{\mathrm{ff}} B_{kl}^{\mathrm{ft}} + G_{kl}^{\mathrm{ff}} G_{kl}^{\mathrm{ft}}\right)\cos(\delta_k - \delta_l) + \left(B_{kl}^{\mathrm{ft}} G_{kl}^{\mathrm{ff}} - B_{kl}^{\mathrm{ff}} G_{kl}^{\mathrm{ft}}\right)\sin(\delta_k - \delta_l)\right) \leq (I_{kl}^{\max})^2. \end{aligned} \tag{B.7}$$

To obtain the bound for I_{kl}^{t}, we can interchange k and l, since $I_{kl}^{\mathrm{t}} = -I_{lk}^{\mathrm{f}}$.

[1] Equivalently, we could have a single constraint of the form $\max\{I_{kl}^{\mathrm{f}}, I_{kl}^{\mathrm{t}}\} \leq I_{kl}^{\max}$, however this constraint is not differentiable everywhere, therefore we opt for two separate constraints.

C

Modified 118 Bus Network

In Chapter 9a, calculations are performed on a modified version of the *IEEE 118 bus* model. In this appendix we describe the modifications that were made to the model. Figure C.1 shows a single line diagram of the grid.

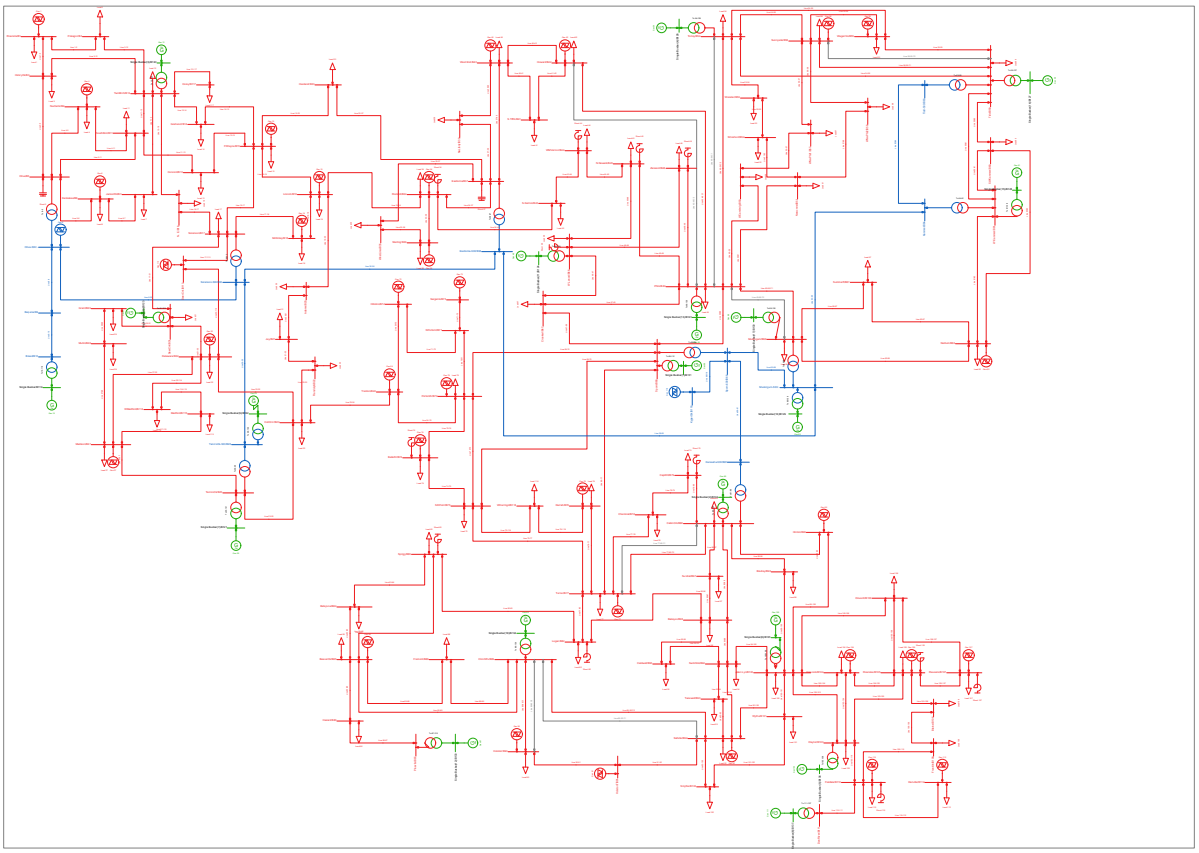

Figure C.1: Single line diagram of the 118 bus network.

The original model was downloaded as a PowerFactory model from [43], and the model is described in [24]. We chose this model because it had a reasonable size, a PowerFactory model was available, and the grid only included basic devices, such as generators, loads, transformers, and shunts. The model was originally constructed to study the transient behaviour of power systems.

The model included all necessary data to be able to perform a Power Flow calculation, such as injections and set points of all loads and generators, loading limits for edges and tap settings for transformers.

Edge name	# parallel edges
Line 4-5	3
Line 8-9	2
Line 9-10	2
Line 15-17	2
Line 23-25	2
Line 25-27	2
Line 34-37	2
Line 42-49 C1	2
Line 42-49 C2	0
Line 49-54 C1	0
Line 49-54 C2	2
Line 49-66 C1	3
Line 49-66 C2	0
Line 56-59 C1	2
Line 56-59 C2	0
Line 77-80 C1	0
Line 77-80 C2	2
Line 89-90 C1	2
Line 89-90 C2	0
Line 89-92 C1	0
Line 89-92 C2	2
Line 100-103	2

Table C.1: Number of times each edge was duplicated. A 0 indicates that the edge was removed.

However, when performing a power flow calculation, some edges were very much overloaded, in some cases by more than 300%. Performing an OPF calculation with this data would in most cases lead to an infeasible problem, unless a big part of the injections are controllable. Therefore, we made some modifications to the grid, so that only a few edges were overloaded, and all edges were loaded below 115%.

We made the following modifications.

1. For all lines at the 500 kV level, the rated current[1], was changed from 0.115 kA to 0.4 kA.
2. For all transformers between a 230 kV and a 500 kV node, the rated power[2] was changed from 100 MVA to 500 MVA.
3. For all transformers, the short circuit voltage[3] was set to 3%.
4. The active power injection of generator *Gen 89* was changed from 607 mW to 450 mW.
5. The tap position of all transformers was set to 0.
6. Some edges were removed, while others were duplicated[4] (resulting in parallel edges). The details of this can be seen in Table C.1.

C.1. Modification in Section 8.3

For the calculation in Section 8.3 we required a grid with parallel transformers. Therefore, we made the following additional modifications.

1. For all transformers between a 230 kV and a 500 kV node, the rated power[5] was changed to 250 MVA (intead of 500 MVA).

[1] Attribute `t:sline` in PowerFactory
[2] Attribute `t:strn` in PowerFactory
[3] Attribute `t:uktr` in PowerFactory
[4] Duplicating the line was done using the `e:nlnum` attribute in PowerFactory.
[5] Attribute `t:strn` in PowerFactory

2. All transformers between a 230 kV and a 500 kV were duplicated[6], resulting in 9 pairs of parallel transformers.

[6]Duplicating the transformer was done using the `e:ntnum` attribute in PowerFactory.