Master Thesis

Master's Degree in Electric Power Systems and Drives

Integration of an AC-OPF solver in GridCal

January 2024

Author: Carlos Alegre Aldeano

Supervisors: Marc Cheah Mañé

Josep Fanals i Batllori

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Abstract

The AC-OPF is a topic that has been the focus of many power systems research, since finding the optimal operating point of an electrical grid can translate into significant economic saving for the grid operators at all levels of distribution. It encapsulates the economic dispatch, by the means of a cost objective function, with all the technical constraints that the grid has to respect. The AC-OPF is the most complete formulation of the problem, but it is also computationally expensive.

The work developed in this thesis presents an AC-OPF solver integrated in GridCal, a power system analysis tool developed by the Spanish TSO, Redeia. It has been part of the SIROCO project, which aimed to improve this analysis tool as a complete toolbox for the Spanish TSO. The solver has been developed in Python, based on the Matpower formulation, and adds important features such as transformer setpoints optimization, DC links, reactive power limitations and some improvements on Matpower's solver to improve convergency.

Resumen

Resum

Contents

List of Figures

List of Tables

Glossary

Symbols

C	Capacitance
C_f	Shunt filter capacitance
C_s	Corrected series filter capacitance
$C_s^{'}$	Series filter capacitance
f_c	Cutoff frequency
i_c	Shunt current
i_{cs}	Series converter current
i_{cf}	Shunt Converter current
i_L	Load current
i_s	Source Voltage
L	Inductance
L_f	Shunt filter inductance
L_l	Line inductance
L_s	Corrected series filter inductance
$L_s^{'}$	Series filter inductance
N_T	Series transformer turns ratio
P	Active power
pf	Power factor
P_L	Load active power
Q	Reactive power
Q_L	Load reactive power
Q_{sh}	Shunt converter reactive power
R_f	Shunt filter resistance
R_l	Line resistance
R_s	Corrected series filter resistance
$R_s^{'}$	Series filter resistance
$\underline{S}_L$	Load apparent power
u_c	Series correction voltage
$u_{c,lim}$	Voltage correction limit
u_{cs}	Series converter voltage
u_{cf}	Shunt Converter voltage
u_{DC}	DC Link Voltage
u_G	Grid voltage
u_L	Load voltage
u_s	Source voltage
x_i	Dummy variable

$\underline{x}_i$	Complex dummy variable
$\lvert\underline{x}_i\rvert$	Magnitude of dummy variable
$x_{i,\mathbb{R}}$	Real component of dummy variable
$x_{i,\mathbb{I}}$	Imaginary component of dummy variable
$x_{i,\alpha}$	Alpha component of dummy variable
$x_{i,\beta}$	Beta component of dummy variable
$x_{i,a}$	Phase a component of dummy variable
$x_{i,b}$	Phase b component of dummy variable
$x_{i,c}$	Phase c component of dummy variable
θ_{sys}	System angle
ω_{sys}	System frequency

Acronyms

AC	Alternating Current
CITCEA	Centre d'Innovació Tecnològica en Convertidors Estàtics i Accionaments
DC	Direct Current
DSTATCOM	Distribution Static Synchronous Compensator
DVR	Dynamic Voltage Regulator
FACTS	Flexible AC Transmission Systems
IEEE	Institute of Electrical and Electronics Engineers
LC	Inductive and Capacitive Low Pass Filter
OLTC	On Load Tap Changer
PI	Proportional Integral
PLL	Phase Locked Loop
PSVD	Positive Sequence Voltage Detector
PV	Photovoltaic
RMS	Root Mean Square
SOGI	Second Order Generalized Integrator
UPC	Universitat Politècnica de Catalunya
UPFC	Unified Power Flow Controller
UPQC	Unified Power Quality Conditioner
VRE	Variable Renewable Energy

Preface

1 Introduction

1.1 Background

The Optimal Power Flow (OPF) is regarded as a complicated mathematical problem of upmost importance for grid operators. While a grid could operate in very varied conditions, the goal is to pick an optimal operating point that minimizes a given objective function and at the same time respects a set of technical constraints.

As an originally non-convex hard problem, the OPF can take many forms. For instance, an economical dispatch would be the simplest variation in which the power flows are dismissed; the DCOPF considers line flows but only solves for the voltage angles; whereas the ACOPF follows the purest formulation using the complete formulation of the problem. Both of them have their strong and weak points, exchanging computational speed for precision (and even feasibility) of the solution. [**chatzivasileiadis2018optimization**].

Regarding a more technical approach, one would consider the AC-OPF to be the best choice without loosing the ability to perform economic analysis. The straight forward approach of the basic AC-OPF problem is Matpower, an open source package for Matlab that allows the user to perform simulations with specific grid models. It is regarded as the baseline of the AC-OPF formulation for many works in the topic. The solution found, due to the non-convex nature of the problem, is a local optimal point.

There have been many development lines that try to find a global optimal point using relaxations over the constraints of the problem in such a way that the resulting transformed problem is convex where it is posible to find the global optimal point using Interior Point Methods (IPMs from now on). These relaxation methods, unlike the DC aproximations, expand the feasibililty domain of solutions instead of modifying it as seen in FIGURE 1

1.2 Objectives

The aim of this project, which is part of the Redeia (Spanish TSO) SIROCO project, is the implementation of an AC-OPF in GridCal, a Python grid analysis package in development by Redeia's engineer Santiago Peñate Vera. In Figure 1 we can see the plot in GridCal's GUI of the Spanish grid. This grid has been provided by Redeia, and it has been one of the many models that have been tested throughout the project, although this particular model's data cannot be showcased.

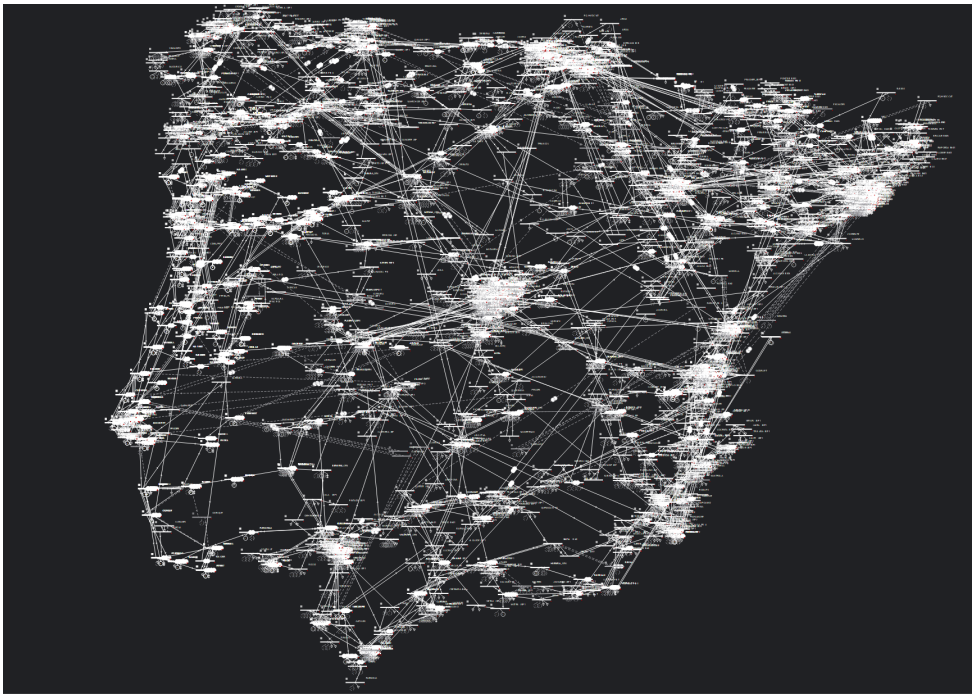

Figure 1: Plot of the topology of the Spanish grid in GridCal interface.

The objective will be to be able to solve the AC-OPF problem for similar grids and be able to perform different studies depending on the user's needs. Figure 2 shows the resulting product of the AC-OPF solver in the GUI. For grids of this size, it is obvious that one should use an appropiate tool to visualize the results with detail, which is the purpose of the GridCal GUI. In Chapter 5, detailed results for some of the study cases will be shown to demonstrate the developed solver and GridCal capabilities.

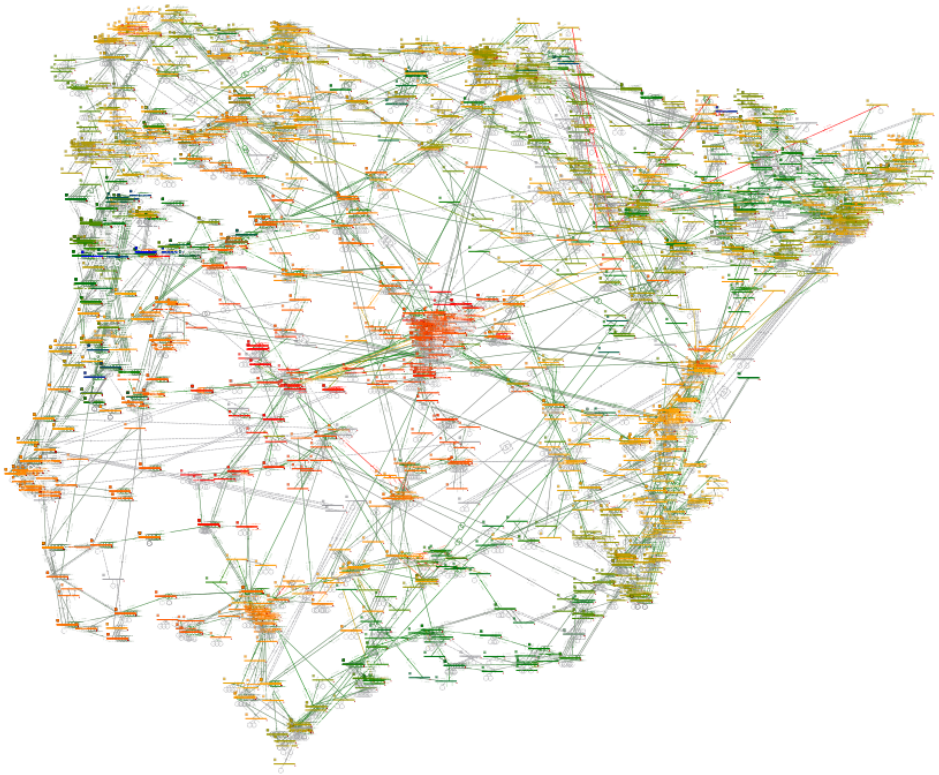

Figure 2: Plot of the Spanish grid solved with the AC-OPF solver in GridCal interface.

The starting point of this tool will be the Matpower formulation, which will be adapted to the GridCal environment with the objective of integrating it and expanding its functionalities. The first steps will be directed to understanding the formulation and incorporate the solver in Python, adapting the initialization of the problem to the GridCal models, which can be obtained from the matpower models using a built-in parser. During this process, the formulation has to be as general as possible to allow the introduction of new features.

Once the solver is operative, the new features will be added to the model. These features will be optional and selectable from GridCal's GUI. The connection of this software with the GUI is out of the scope of this project, as it will be done by a third party. The additional features are the possibility to optimize controlled transformer setpoints, simple DC links models, reactive generation limitations and the addition of slack variables for operational range that can be violated in exchange of a penalty cost. This last feature improves (and sometimes even allows) the convergence of some grid states that are difficult to solve.

On top of that, the resulting algorithm will be prepared to introduce new features already being discused such as a complete AC/DC modelling, better control models and even the addition of relaxations in case faster studies are required.

1.3 Outline

Chapter 1 will describe all the elements of an electrical grid, the parameters that define each of them and the notation used throughout the thesis. The relationships between elements and the physics that determine the operation of an electrical grid are described in Chapter 2. It will be followed in Chapter 3 by the mathematical modelling of the optimization problem that has to be solved to determine optimal operation. Chapter 4 contains all the software structure and algorithms developments, accompanied by some explanations of the GridCal environment. In Chapter 5, benchmarking cases will be shown for each implemented feature.

2 Grid Elements

This section describes the elements and information contained in the grid model. The two elements that shape the networks are the buses and lines, which can be also named nodes and branches using a topological naming. On top of some of the buses we can find generators and loads, and on top of some of the lines there can be transformers.

All of these elements have associated information about their working parameters, regarding operation limits, element subtype, monitoring information and other relevant information. Every piece of information needed to run the optimization program over the grid is described in this section.

2.1 Buses

The buses, or the nodes of the network, are the points on the grid where generation or consumption of electric power occurs. The generation comes from generators connected to the bus, while the consumption, often refered to as load, comes from the sum of all the users of electrical devices connected to the bus.

When dealing with a transport grid, buses can be seen as the substations that feed an entire neighborhood, town, or even city. While there is a whole distribution network behind each bus, they are modelled as single points that aggregate all the generation and consumption beneath them. It will be shown later that larger generator units are treated individually since their operating points are typically dictated by the TSO and they have the ability to affect the operation of the grid.

The set of buses of the electrical grid is denoted as $N = \{1, ..., i, ..., j, ..., n\}$, where the index i is used for the $from$ bus, while j is used for the to bus when dealing with an interaction of two buses. Each bus has a state represented by a 2 complex variables, the complex voltage $v_i \angle \theta_i$ and the net apparent power $S_i = P_i + jQ_i$.

The voltage magnitude is expressed in per unit, each grid having its own reference voltage. The phase, expressed in radians, corresponds to the angular difference with respect to the reference bus, also known as the *slack* bus, which is assigned a phase value of 0. The real part of the net apparent power corresponds to the active power, while the imaginary part corresponds to the reactive power. This net power is the balance of the bus's generation and consumption. It must match the power exchange with other buses to accomplish the nodal power balance, which imposes that the net power of the bus should be 0 in equilibrium.

There are several types of bus depending on which variables can be controlled. The most common ones, and the ones that will be considered in the modelling of the grid, are the following:

- The *slack* bus, or reference bus, serves as a voltage magnitude and angle reference. As such, its voltage variables are set as $V_i \angle \theta_i = 1 \angle 0°$, using per unit. This type of bus can be regarded as the balancing bus, as it will provide the power necessary to solve any power imbalance when the power flow is solved. Generally, they are the bus with the greatest and most stable generation capacity.

  Since there can be more than one slack bus in a grid due to being different electrical islands, there will be a list of $nslack$ buses called $slack$. It is important to note that each island has to have its own slack bus, otherwise the power flow will not be solvable.

- The PQ buses have known power exchange values. The P_i and Q_i values are forecasted for all the simulation runtime. They normally are buses where there is no generation, and the exchange power is directly given by the demand forecast. The voltage magnitude and angle are unknowns the value of which will be obtained after solving the power flow.

  The list of PQ buses is denoted as pq and has npq elements. Slack buses can be included in this list, and they will only have a fixed voltage phase of 0. The buses that will fit in this list are the ones with different upper and lower voltage limits.

- The PV buses have their active power exchange P_i, and their voltage module v_i as controlled magnitudes. Their values are setpoints determined by their operator The reactive power generation and the phase of the bus are unknowns the value of which is obtained after solving the power flow.

  The list of PV buses is denoted as pv and has npv elements. Same as with the PQ buses, slack buses can be included in this list. The buses with equal upper and lower voltage limits will be included in this list, and will have that value as the voltage setpoint.

The goal of the optimization will to obtain the optimal setpoints of this controlled buses, as it will be shown in the optimization formulation.

2.2 Lines

The electrical lines connect two buses and allows the transportation of power between them. The power transfer can happen in both directions, although obviously only one at a time. The line has two ends, the 'from' end f and the 'to' end t. The lines are identified with the index of each bus at each position and will remain the same independently of the direction of the flow. The sign of the power will be positive when going in the direction 'from' $\rightarrow$ 'to', and negative in the direction 'to' $\rightarrow$ 'from'.

The set of lines is denoted as $M = \{1, ..., k, ..., m\}$, while the set of 'from' and 'to' buses is denoted as $F = \{f_1, ..., f_m\}$ and $T = \{t_1, ..., t_m\}$ respectively. A line is defined used the notation

$k \triangleq (f_k, t_k)$. This line will be modelled as a π-equivalent line, which will mean that the power transfer will be calculated using the resulting admittance expression and the variables of each bus. The power calculation is derived in following sections.

The lines have a limit over the power they can withstand. This limit, called line rate, has to be checked at both ends of the line, since it is not equal due to line losses. The lines that are monitored and will have this rate applied are the ones with the monitor loading flag set to 1. The list is obtained from the GridCal model of the grid, and will be stored in the il list of indexes containing nll monitored branch elements.

2.3 Transformers

The transformers change the level of the voltage between both of its ends. The voltage when going from generation towards the transport grid raises to lower loses, and when arriving to consumption areas lowers again to reduce the risks of manipulation. In the model presented, the transformers are modelled as lines with a transformation of the voltage level. Since per unit is used, this change does not directly imply modification of the voltage module value, but when recovering the actual value in Volts, each bus will have its own nominal voltage level.

In addition to this, the transformers' output voltages are considered adjustable in both module and phase using the called tap variables. Tap module m_p corresponds to the deviation, in per unit, from the nominal transformation ratio, while tap angle τ, in radians, corresponds to the shift added to the voltage phase. The majority of the transformers are considered not controlled, but some of them can be controlled in one or both variables. For that reason, their setpoints can be added as variables of the optimization problem.

The indexing lists containing the branches with controlled transformers is $\boldsymbol{k_m}$ for module controlled transformers and $\boldsymbol{k_{tau}}$ for phase controlled transformers. Those who have controls for both variables will be in both lists.

3 Grid Model

Once all the elements of the grid have been defined, the model of the power flow can be properly explained. The approach chosen is aligned with the models proposed by GridCal, which make use of the Universal Branch Model to describe the admittance matrices of the branches. The power flow model used during this project has been the Bus Injection Model (BIM), which will be calculated at each bus of the grid, and has to be equal to zero for the system to be in equilibrium. This model relates voltages, power transfer, generation and demand, which are the variables of the optimization problem.

The notation used throughout this work is as follows: vectors are denoted in bold (can be both

upper or lowercase), matrices are denoted in uppercase, and scalars are denoted in lowercase. Vectors that are transformed in diagonal matrices appear within square brackets. Vectors that slice other vectors using their values as indexes appear after the slice vector within curly brackets.

3.1 Power Flow Equations

We will show the construction of the power flow equations, since some of the intermediate steps can be useful for later purposes. We start from the power flowing through a branch using the Universal Branch Model for the k branch:

$$\begin{bmatrix} i_f \\ i_t \end{bmatrix} = \begin{bmatrix} y_{ff} & y_{ft} \\ y_{tf} & y_{tt} \end{bmatrix} \begin{bmatrix} v_f \\ v_t \end{bmatrix} = Y_{br} \begin{bmatrix} v_f \\ v_t \end{bmatrix} \tag{1}$$

This equation relates the voltages of the buses connected by the branch and the currents flowing from them. The branch admittance matrix terms can be calculated using the parameters of the branch as follows:

$$Y_{br} = \begin{bmatrix} (y_s + j\frac{b_c}{2})\frac{1}{m_p^2} & -\frac{y_s}{m_p e^{-j\tau}} \\ -\frac{y_s}{m_p e^{j\tau}} & y_s + j\frac{b_c}{2} \end{bmatrix} \tag{2}$$

where y_s is the series admittance calculated as $y_s = \frac{1}{R+jX}$, b_c is the capacitance of the line, m_p is the transformer ratio, and τ is the phase shift of the transformer.

The expression above contains the information of a single line. In the grid model used, $y_f f$, $y_f t$, $y_t f$ and $y_t t$ are vectors of length m that store these primitives of the admittance matrix for all the branches in the model.

Consider the connectivity matrices C_f and C_t, with size kxn that relate the index of the 'from' and 'to' buses to the index of their branch as follows:

$$C_{f_{ki}} = \begin{cases} 1, & \text{for } k == (i,j) \quad \forall j \in \boldsymbol{N} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

$$C_{t_{kj}} = \begin{cases} 1, & \text{for } k == (i,j) \quad \forall i \in \boldsymbol{N} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Using these connectivity matrices and the primitives, $from$ and to admittance matrices can be constructed. The to admittance matrix contains the elements of the admittance matrix that are connected to the 'to' bus of the branch, while the $from$ admittance matrix contains the elements connected to the 'from' bus of the branch. The bus admittance matrix Y_{bus}, which contains all the information needed to calculate the power flow, can then be constructed.

$$\begin{aligned} Y_f &= [\boldsymbol{Y_{ff}}]\, C_f + [\boldsymbol{Y_{ft}}]\, C_t \\ Y_t &= [\boldsymbol{Y_{tf}}]\, C_f + [\boldsymbol{Y_{tt}}]\, C_t \\ Y_{bus} &= C_f^\top Y_f + C_t^\top \boldsymbol{Y_t} + [\boldsymbol{Y_{sh}}] \end{aligned} \tag{5}$$

where the brackets indicate that the vector is transformed into a diagonal matrix and the $[\boldsymbol{Y_{sh}}]$ matrix contains the shunt admittance of each bus, which is a given parameter of the model. The resulting matrices have size *m x n*.

The model used to calculate the power balance of the system is the bus injection model, which is based on the equality between the power injected in a bus and power flowing out of it. The injection can come from generators connected at the bus or lines from buses while the outflow goes to demand connected to the bus or lines that transport energy to other buses.

The vector of complex power injections from lines $\boldsymbol{S_{bus}}$ is given by the following equations:

$$\boldsymbol{S_{bus}} = [\boldsymbol{V}](Y_{bus}\boldsymbol{V})^* \tag{6}$$

where $\boldsymbol{V}$ is the vector of the complex voltages of the buses calculated as $\boldsymbol{\mathcal{V}}e^{j\boldsymbol{\theta}}$ using element to element multiplication. Again, the notation $[\boldsymbol{V}]$ refers to the diagonal matrix created from the vector $\boldsymbol{V}$. The complete balance can be now calculated as:

$$\boldsymbol{G^S_{bus}} = \boldsymbol{S_{bus}} - \boldsymbol{S^g_{bus}} + \boldsymbol{S^d_{bus}} = \boldsymbol{0} \quad \text{(equilibrium)} \tag{7}$$

The notation $\boldsymbol{G^S}$ will be useful later to refer to this balance as an equality constraint of the optimization problem.

One particularity to be considered is that there could be multiple generators connected to the same bus, and each of them has to be considered separately since there could be differences in price or operational limits. To do so, $\boldsymbol{S^g}$ is obtained through the use of the *n x ng* connectivity matrix C_G:

$$\boldsymbol{S^g_{bus}} = C_G \boldsymbol{S^g} \tag{8}$$

For buses that are part of a DC link, the power balance has to account for the power transferred through the link.

$$\begin{aligned} \boldsymbol{G^S_{bus}}\{\boldsymbol{fdc}\} &+= \boldsymbol{P_{dc}}\{\boldsymbol{dc}\} \\ \boldsymbol{G^S_{bus}}\{\boldsymbol{tdc}\} &-= \boldsymbol{P_{dc}}\{\boldsymbol{dc}\} \end{aligned} \tag{9}$$

In addition to the bus power balance, the line power balance has to be considered to ensure that operational limits of the lines are not exceeded. The power flowing through a line can be calculated using the $from$ and to admittance matrices and the voltages of the buses connected to the line. Both sides of the lines have to be considered, since the powers at the ends are not equal and the direction of the flow can change depending on grid conditions.

$$\begin{aligned} \boldsymbol{V_f} &= C_f \boldsymbol{V} \\ \boldsymbol{V_t} &= C_t \boldsymbol{V} \\ \boldsymbol{S_f} &= [\boldsymbol{V_f}](Y_f \boldsymbol{V})^* \\ \boldsymbol{S_t} &= [\boldsymbol{V_t}](Y_t \boldsymbol{V})^* \end{aligned} \tag{10}$$

Here it is important to note that the vector $\boldsymbol{V_f}$ will contain the voltage value of the $from$ bus of each line, which means there can be some buses appearing more than one time.

3.2 Operational limits

Each element of the grid has its owns operational limits that have to be respected. The limits are stored in lists of length equal to the respective element count. These limits correspond to inequalities of the optimization problem. The notation $\boldsymbol{H^I}$ is used to identify to each different limit.

3.2.1 Nodal voltage limits

The voltage module limits of the buses are stored in the $\boldsymbol{V_{max}}$ and $\boldsymbol{V_{min}}$ arrays, with its value in per unit. The limits are checked using the following equations:

$$\begin{aligned} \boldsymbol{H^{v_u}} &= \boldsymbol{\mathcal{V}} - \boldsymbol{V_{max}} \leq \boldsymbol{0} \quad \text{(upper limit)} \\ \boldsymbol{H^{v_l}} &= \boldsymbol{V_{min}} - \boldsymbol{\mathcal{V}} \leq \boldsymbol{0} \quad \text{(lower limit)} \end{aligned} \tag{11}$$

3.2.2 Generation limits

The generation limits apply to both the active and reactive power of the generators. They are stored in the P_{max}, P_{min}, Q_{max} and Q_{min} lists, with its value in per unit. The limits are checked using the following equations:

$$\begin{aligned} \boldsymbol{H}^{P_u} &= \boldsymbol{P_g} - \boldsymbol{P_{max}} \leq \mathbf{0} \quad \text{(upper limit)} \\ \boldsymbol{H}^{P_l} &= \boldsymbol{P_{min}} - \boldsymbol{P_g} \leq \mathbf{0} \quad \text{(lower limit)} \\ \boldsymbol{H}^{Q_u} &= \boldsymbol{Q_g} - \boldsymbol{Q_{max}} \leq \mathbf{0} \quad \text{(upper limit)} \\ \boldsymbol{H}^{Q_l} &= \boldsymbol{Q_{min}} - \boldsymbol{Q_g} \leq \mathbf{0} \quad \text{(lower limit)} \end{aligned} \tag{12}$$

3.2.3 Transformer limits

The controllable transformers will have their limits for the controllable variables limited. THe limits are similar to the previous:

$$\begin{aligned} \boldsymbol{H}^{m_{p_u}} &= \boldsymbol{m_p} - \boldsymbol{m_{p_{max}}} \leq \mathbf{0} \quad \text{(upper limit)} \\ \boldsymbol{H}^{m_{p_l}} &= \boldsymbol{m_{p_{min}}} - \boldsymbol{m_p} \leq \mathbf{0} \quad \text{(lower limit)} \\ \boldsymbol{H}^{\tau_u} &= \boldsymbol{\tau} - \boldsymbol{\tau_{max}} \leq \mathbf{0} \quad \text{(upper limit)} \\ \boldsymbol{H}^{\tau_l} &= \boldsymbol{\tau_{min}} - \boldsymbol{\tau} \leq \mathbf{0} \quad \text{(lower limit)} \end{aligned} \tag{13}$$

3.2.4 Line limits

The line limits will be checked considering the rating of each line in per unit. Since the direction of the flow is not known, and to avoid using absolute values, this constraint is considered quadratically.

$$\begin{aligned} \boldsymbol{H}^{S_f} &= \boldsymbol{S_f}\boldsymbol{S_f^*} - \boldsymbol{S_{max}^2} \leq \mathbf{0} \\ \boldsymbol{H}^{S_t} &= \boldsymbol{S_t}\boldsymbol{S_t^*} - \boldsymbol{S_{max}^2} \leq \mathbf{0} \end{aligned} \tag{14}$$

3.2.5 Bound slack variables

So far, all the constraints described has physical meaning and are related to the grid model. The last set of constraints are related to the optimization problem and are used to improve, or even make possible, the convergence of the solver. The bound slacks are used to allow the solution to surpass some operational limits in order to find a feasible solution. Of course, it should come at a cost and the objective function will be used to penalize this behavior.

These bound slacks are applied to the voltage limits and to the line limits, allowing slight over or undervoltages and some overloads. The constraint equations are modified and the slack variables are imposed to be positive. The equations are as follows:

$$
\begin{aligned}
\boldsymbol{H}^{v_u} &= \boldsymbol{\mathcal{V}} - \boldsymbol{V_{max}} - \boldsymbol{sl}_{v_{max}} \leq \mathbf{0} \quad \text{(upper limit)} \\
\boldsymbol{H}^{v_l} &= \boldsymbol{V_{min}} - \boldsymbol{\mathcal{V}} - \boldsymbol{sl}_{v_{min}} \leq \mathbf{0} \quad \text{(lower limit)} \\
\boldsymbol{H}^{S_f} &= \boldsymbol{S_f}\boldsymbol{S_f^*} - \boldsymbol{S_{max}^2} - \boldsymbol{sl_{sf}} \leq \mathbf{0} \quad \text{(from limit)} \\
\boldsymbol{H}^{S_t} &= \boldsymbol{S_t}\boldsymbol{S_t^*} - \boldsymbol{S_{max}^2} - \boldsymbol{sl_{st}} \leq \mathbf{0} \quad \text{(to limit)} \\
-\boldsymbol{H}^{sl_{v_{max}}} &\leq \mathbf{0} \\
-\boldsymbol{H}^{sl_{v_{min}}} &\leq \mathbf{0} \\
-\boldsymbol{H}^{sl_{sf}} &\leq \mathbf{0} \\
-\boldsymbol{H}^{sl_{st}} &\leq \mathbf{0}
\end{aligned}
\tag{15}
$$

The use of this bound slacks is optional and can be removed from the optimization problem if the user prefers to have a solution in compliance with the established limits.

4 Optimization Problem

This section describes the optimization model used to solve the AC Optimal Power Flow problem. The mathematical formulation of this problem is presented using abstract notation of the KKT conditions for optimization problems, which later on will regain a more practical form. The implementation of the solver is also described, as well as the Python code used to solve the problem. The general theory of optimization can be found in the book of Boyd and Vandenberghe [**boyd2004convex**], in Nocedal and Wright, and the adaptation of this theory to a more compact formulation can be reviewed in the Matpower Interior Point Solver documentation.

4.1 Optimization problem

The general optimization problem involving equality and inequality constraints can be formulated as in Equation 16.

$$
\begin{aligned}
\min \quad & f(\boldsymbol{x}) \\
\text{s.t.} \quad & \boldsymbol{G}(\boldsymbol{x}) = \mathbf{0} \\
& \boldsymbol{H}(\boldsymbol{x}) \leq \mathbf{0}
\end{aligned}
\tag{16}
$$

where $\boldsymbol{x} \in \mathbb{R}^n$, being n the number of decision variables, $f(x)$ is the function to be minimized which typically accounts for the generation cost, $\boldsymbol{G}(\boldsymbol{x})$ is a set of equality constraints of the

problem, and $\boldsymbol{h}(\boldsymbol{x})$ contains the technical restrictions to respect voltage and line flow limits. The bold notation is used to indicate vectors.

The problem will be solved using the barrier parameter method, which consists of solving a sequence of optimization problems with different values of the barrier parameter γ. The barrier parameter is an interior point method used to penalize the violation of the inequality constraints in the problem. This algorithm is used to solve the problems by moving through the interior of the feasible region.

The lagrangian of the problem is defined by Equation **??**

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{Z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^\top \boldsymbol{G}(\boldsymbol{x}) + \boldsymbol{\mu}^\top (\boldsymbol{H}(\boldsymbol{x}) + \boldsymbol{Z}) - \gamma \sum_{i=1}^{n_i} \log(z_i) \tag{17}$$

where $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are the Lagrange multipliers associated with the equality and inequality constraints respectively.

The KKT conditions are a set of equations that must be satisfied by the solution of the optimization problem. These conditions are a result of imposing that the partial derivatives of the Lagrangian in the (x, Z, λ, μ) variable space equals 0, as shown in Equation 18.

$$\begin{aligned}
\nabla_{x,Z,\lambda,\mu} \mathcal{L}(\boldsymbol{x}, \boldsymbol{Z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \mathbf{0} \\
\boldsymbol{\mathcal{L}_x}(\boldsymbol{x}, \boldsymbol{Z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) &= f_x + \boldsymbol{\lambda}^\top G_x + \boldsymbol{\mu}^\top H_x = \mathbf{0} \\
\boldsymbol{\mathcal{L}_Z}(\boldsymbol{x}, \boldsymbol{Z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) &= \boldsymbol{\mu}^\top - \gamma \mathbf{1}_{n_i}^\top [\boldsymbol{Z}]^{-1} = \mathbf{0} \\
\boldsymbol{\mathcal{L}_\lambda}(\boldsymbol{x}, \boldsymbol{Z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) &= \boldsymbol{G}^\top(\boldsymbol{x}) = \mathbf{0} \\
\boldsymbol{\mathcal{L}_\mu}(\boldsymbol{x}, \boldsymbol{Z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) &= \boldsymbol{H}^\top(\boldsymbol{x}) + \boldsymbol{Z}^\top = \mathbf{0}
\end{aligned} \tag{18}$$

where $\mathbf{1}_{n_i}$ is a vector of ones of length n_i, and γ is the barrier parameter that will be updated in each iteration of the solver until reaching a value close to zero. The notation $\{_{\boldsymbol{X}}$ indicates the gradient of the given function with respect to the vector $\boldsymbol{X}$.

The first gradient corresponds to the stationary condition of the problem, the second gradient corresponds to the complementary slackness, and the other two gradients correspond to the primal feasibility conditions of the problem. These conditions, alongside the dual feasibility condition $Z_i, \mu_i \geq 0 \quad \forall i$, are necessary and sufficient for the solution of the optimization problem. The resulting set of equations will be solved computationally fue to the non-convex nature of the problem. For this type of general problem, the approach chosen is the use of an Interior Point Method (IPM) solver.

4.2 Interior Point Method solver

The choice of this type of solver is regarded as the best option to deal with non-linear optimization, as shown in the dedicated chapter 19 in Nocedal and Wright's book. The IPM solver is a type of optimization algorithm that solves the KKT conditions of the problem iteratively. There are multiple algorithms that can be used, although the one presented in this project is the Newton-Raphson. In reference we this algorithm is compared with the Gauss-Seidel algorithm, yielding better scalability for the Newton-Raphson method. Since this algorithm will be used by the Spanish TSO, the scalability of the solver is a key factor to consider. There can be improvements to this method as presented in the literature, but for the purpose of this project, the IPM has been implemented directly using the Newton-Raphson method with a step-control mechanism.

The Newton-Raphson iterative process that will find the roots of the KKT conditions can be described as in Algorithm 1.

Algorithm 1 Newton-Raphson Iterative Process

1: Initialize $\boldsymbol{x}_0$, tolerance ϵ, and set $k = 0$
2: **while** $||\boldsymbol{f}(\boldsymbol{x}_k)|| > \epsilon$ **do**
3: Compute the Jacobian matrix $J(\boldsymbol{x}_k)$
4: Solve $J(\boldsymbol{x}_k)\Delta\boldsymbol{x}_k = -\boldsymbol{f}(\boldsymbol{x}_k)$ for $\Delta\boldsymbol{x}_k$
5: Apply step control to $\Delta\boldsymbol{x}_k$
6: Update $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta\boldsymbol{x}_k$
7: $k = k + 1$
8: **end while**

The algorithm will perform several Newton-Raphson steps until the convergence criteria is met, meaning that equations 18 are solved. The structure of this algorithm substituting the submatrices present at the jacobian matrix is presented in Equation 19

$$-\begin{pmatrix} \boldsymbol{\mathcal{L}_{xx}} & \mathbf{0} & \boldsymbol{G_x^\top(x)} & \boldsymbol{H_x^\top(x)} \\ \mathbf{0} & [\boldsymbol{\mu}] & \mathbf{0} & [\boldsymbol{Z}] \\ \boldsymbol{G_x(x)} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{H_x(x)} & \boldsymbol{I} & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\Delta x} \\ \boldsymbol{\Delta Z} \\ \boldsymbol{\Delta\lambda} \\ \boldsymbol{\Delta\mu} \end{pmatrix} = \begin{pmatrix} \mathcal{L}_x^\top \\ [\boldsymbol{\mu}]\boldsymbol{Z} - \gamma\mathbf{1}_{n_i} \\ \boldsymbol{G(x)} \\ \boldsymbol{H(x)} + \boldsymbol{Z} \end{pmatrix} \tag{19}$$

where $\boldsymbol{\Delta x}$ represents the variation of $\boldsymbol{x}$ (and similarly to $\boldsymbol{s}$, $\boldsymbol{y}$, $\boldsymbol{z}$), $\boldsymbol{I}$ is the identity matrix, and the term $\boldsymbol{\nabla^2_{xx}\mathcal{L}}$ is the derivative of the first KKT condition, which was already the gradient of the Lagrangian of the problem.

Note that the last of the KKT conditions is not included in the system of equations. This condition will be enforced by the algorithm when choosing the step length of the iteration when

updating the variables.

This system of equations can be further reduced using methods that can be found in Nocedal and Wright's book, derived with detail in the MIPS Manual. The relevant equations obtained during the reduction process are shown in Equations 20 and 21.

$$\begin{aligned}
[\boldsymbol{\mu}]\Delta\boldsymbol{Z} + [\boldsymbol{Z}]\Delta\boldsymbol{\mu} &= -[\boldsymbol{\mu}]\boldsymbol{Z} + \gamma\mathbf{1}_{n_i} \\
[\boldsymbol{Z}]\Delta\boldsymbol{\mu} &= -[\boldsymbol{Z}]\boldsymbol{\mu} + \gamma\mathbf{1}_{n_i} - [\boldsymbol{\mu}]\Delta\boldsymbol{Z} \\
\Delta\boldsymbol{\mu} &= -\boldsymbol{\mu} + [\boldsymbol{Z}]^{-1}(\gamma\mathbf{1}_{n_i} - [\boldsymbol{\mu}]\Delta\boldsymbol{Z}). \\
\boldsymbol{H}_x\Delta\boldsymbol{X} + \Delta\boldsymbol{Z} &= -\boldsymbol{H}(\boldsymbol{X}) - \boldsymbol{Z} \\
\Delta\boldsymbol{Z} &= -\boldsymbol{H}(\boldsymbol{X}) - \boldsymbol{Z} - \boldsymbol{H}_x\Delta\boldsymbol{X}.
\end{aligned} \tag{20}$$

$$\begin{aligned}
\boldsymbol{M} &\equiv \boldsymbol{\mathcal{L}_{xx}} + \boldsymbol{H_x}^\top[\boldsymbol{Z}]^{-1}[\boldsymbol{\mu}]\boldsymbol{H_x} \\
&= \boldsymbol{f_{xx}} + \boldsymbol{G_{xx}}(\boldsymbol{\lambda}) + \boldsymbol{H_{xx}}(\boldsymbol{\mu}) + \boldsymbol{H_x}^\top[\boldsymbol{Z}]^{-1}[\boldsymbol{\mu}]\boldsymbol{H_x} \\
\boldsymbol{N} &\equiv \boldsymbol{\mathcal{L}_x}^\top + \boldsymbol{H_x}^\top[\boldsymbol{Z}]^{-1}(\gamma\mathbf{1_{n_i}} + [\boldsymbol{\mu}]\boldsymbol{H}(\boldsymbol{x})) \\
&= \boldsymbol{f_x}^\top + \boldsymbol{G_x}^\top\boldsymbol{\lambda} + \boldsymbol{H_x}^\top\boldsymbol{\mu} + \boldsymbol{H_x}^\top[\boldsymbol{Z}]^{-1}(\gamma\mathbf{1_{n_i}} + [\boldsymbol{\mu}]\boldsymbol{H}(\boldsymbol{x})).
\end{aligned} \tag{21}$$

The reduced system of equations using this refactoring is shown in Equation 22.

$$\begin{bmatrix} \boldsymbol{M} & \boldsymbol{G_x}^\top \\ \boldsymbol{G_x} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{N} \\ -\boldsymbol{G}(\boldsymbol{x}) \end{bmatrix} \tag{22}$$

As it can be seen, the inequality constraint are not included in the system of equations. The solver will firstly solve this smaller matricial problem to obtain the variation of the decision variables and the Lagrange multipliers associated with the equality constraints. Then, the variation of the inequality constraint multipliers and slack will be found using the expressions derived above.

Before proceeding to update the values of the variables, there will be two additional steps to be taken. The first one is to ensure that the dual feasibility condition

$$\boldsymbol{Z_i}, \boldsymbol{\mu_i} \geq 0 \quad \forall i$$

is met. This will be done by using the maximum step length such that the values of the multipliers remain positive. The calculation of this step length is shown at Equation 23.

$$\alpha_p = \min\left(\tau \cdot \min\left(-\frac{\boldsymbol{Z}}{\Delta \boldsymbol{Z}}\right), 1\right)$$
$$\alpha_d = \min\left(\tau \cdot \min\left(-\frac{\boldsymbol{\mu}}{\Delta \boldsymbol{\mu}}\right), 1\right) \tag{23}$$

where τ is a parameter that will be set to a value really close to 1, without exceeding it (i.e. 0.9995).

This step length will be tested to ensure that it is not too large as it could lead to divergence of the algorithm. This will be done by using a step control mechanism that will reduce the step length if the conditions are not met. The step control mechanism will update just the decision variables x, using the Lagrangian of the system as a reference. The step control mechanism is described in Algorithm 2.

Algorithm 2 Newton-Raphson Iterative Process

Initialize $\mathcal{L}_0 = f(\boldsymbol{x}) + \boldsymbol{\lambda}^\top \boldsymbol{G}(\boldsymbol{x}) + \boldsymbol{\mu}^\top \boldsymbol{H}(\boldsymbol{x}) - \gamma \sum_{i=1}^{n_i} \log(Z_i)$, $\alpha = 1$, use its gradient $\boldsymbol{\mathcal{L}_x}$ and hessian $\boldsymbol{\mathcal{L}_{xx}}$ previously used for the jacobian, and set $\alpha = trust$
while $i < step\_control\_iterations$ **do**
  $\Delta \boldsymbol{x}_1 = \alpha \Delta \boldsymbol{x}$
  $\boldsymbol{x}_1 = \boldsymbol{x} + \Delta \boldsymbol{x}_1$
  Compute $\mathcal{L}_1$ and $\mathcal{L}_{x_1}$
  $\rho = \frac{\mathcal{L}_1 - \mathcal{L}}{\boldsymbol{\mathcal{L}_x} \Delta \boldsymbol{x_1} + 0.5 \Delta \boldsymbol{x_1}^\top \boldsymbol{\mathcal{L}_{xx}} \Delta \boldsymbol{x_1}}$
  if ρ is between ρ_{lower} and ρ_{upper} **then**
    break
  else
    $i = i + 1$
    $\alpha = \alpha / 2$
    print 'Use step control!'
  end if
end while

This will reduce all the step sizes if the new lagrangian does not accomplish the condition. If this condition is not activated, then $\alpha = 1$ Now, the values of all the variables and multipliers can be updated with the final values of the step length as shown in Equation 24.

$$
\begin{aligned}
\boldsymbol{x} &\leftarrow \boldsymbol{x} + \alpha\alpha_p\Delta\boldsymbol{x} \\
\boldsymbol{Z} &\leftarrow \boldsymbol{Z} + \alpha\alpha_p\Delta\boldsymbol{Z} \\
\boldsymbol{\lambda} &\leftarrow \boldsymbol{\lambda} + \alpha\alpha_d\Delta\boldsymbol{\lambda} \\
\boldsymbol{\mu} &\leftarrow \boldsymbol{\mu} + \alpha\alpha_d\Delta\boldsymbol{\mu}
\end{aligned}
\tag{24}
$$

The barrier parameter is updated after this new point is calculated using Equation 25.

$$
\gamma \leftarrow \sigma\frac{\boldsymbol{Z}^\top\boldsymbol{\mu}}{n_i} \tag{25}
$$

where σ is a parameter that will be between 0 and 1. The value of σ will be set to 0.1, as in Matpower, although Nocedal and Wright proposes some different approaches with different values of this parameter when the solver approaches the solution.

Finally, the last step of the solver will be to calculate the convergence criteria to determine if there should be another iteration of the algorithm. There are three criteria that will be used to determine if the algorithm: The feasibility condition, the gradient condition and the gamma value. All of them have to be below a tolerance ϵ to consider the problem solved. The later condition can be checked performing direct comparison, while the former two can be calculated as shown in Equation 26

$$
\begin{aligned}
\text{feascond} &= \frac{\max\left(\left[\|\mathbf{G}\|_\infty, \max(\boldsymbol{H})\right]\right)}{1 + \max\left(\left[\|\mathbf{x}\|_\infty, \|\mathbf{Z}\|_\infty\right]\right)} \\
\text{gradcond} &= \frac{\|\mathcal{L}_\mathbf{x}\|_\infty}{1 + \max\left(\left[\|\lambda\|_\infty, \|\mu\|_\infty\right]\right)}
\end{aligned}
\tag{26}
$$

The algorithm will continue to iterate until the convergence criteria is met, although there is a maximum number of iterations parameter to avoid infinite loops.

4.3 Python implementation

The resulting algorithm is implemented in Python as a stand-alone optimization solver. The solver is capable of solving any kind of optimization problem as long as the objective function, equality constraints, and inequality constraints are provided alongside their gradients and hessians. The user will have to provide a function that returns an object containing this equations, as well as the desired initialization and tolerance values. The objective of this project is to solve

electrical systems, but any other type of system correctly model can also be solved using this solver.

The pseudo-code described in Algorithm 3 will represent the complete algorithm including all the steps. In this project, the function called to calculate the values of the problem equations will be described in the next section.

4.3.1 Dealing with sparsity

A small comment to be made is that the solver will work with sparse matrices. Without entering in much detail, the usage of this type of matrices available in the SciPy library will allow the solver to be more efficient when dealing with the large systems that result when modelling power grids that are composed of thousands of buses and lines.

This type of matrices are used to store only the non-zero elements of the matrix, not wasting memory in storing the zeros. For instance, the Jacobian obtained in the IEEE14 test case has a size of 129 x 129, although only 794 of the elements are non-zero, representing 4.8% of the total elements.

The larger the system, the lower this fraction of non-zero values is. A case of 300 buses used as a benchmark contains 0.3% of non-zero values in the Jacobian. For this reason, the usage of sparse matrices is extremely important to solve country-sized power grids.

Algorithm 3 Pseudo-code for the algorithm

```
Initialize variables: x, mu, lam, z, gamma, n_ineq, tol, iter_counter, max_iter, converged
while not converged and iter_counter < max_iter do
    Evaluate the functions, gradients and hessians at the current iteration
    Compute the transpose of Hx and Gx
    Compose the Jacobian
    Compose the residual
    Find the reduced problem residuals and split them
    Calculate the inequalities residuals using the reduced problem residuals
    if step_control is enabled then
        Compute l0
        Initialize alpha with trust
        for j in range(20) do
            Compute dx1 and x1
            Evaluate the function at x1
            Compute l1 and rho
            if rho is between rho_lower and rho_upper then
                break
            else
                Halve alpha and print 'Use step control!'
            end if
        end for
        Scale dx, dz, dlam, dmu by alpha
    end if
    Compute the maximum step allowed
    Update the values of the variables and multipliers
    Update gamma
    Update fobj, g, h
    Compute the transpose of Hx and Gx
    Compute the norms of g, lam, mu, z
    Compute lx
    Compute feascond, gradcond, error
    Compute z_inv and mu_diag
    Check if the conditions for convergence are met
end while
```

5 AC-OPF

The Optimal Power Flow (OPF) is regarded as a complicated mathematical problem of upmost importance for grid operators. While a grid could operate in very varied conditions, the goal is to pick an optimal operating point that minimizes a given objective function and at the same time respects a set of technical constraints.

As an originally non-convex hard problem, the OPF can take many forms. For instance, an economical dispatch would be the simplest variation in which the power flows are dismissed; the DCOPF considers line flows but only solves for the voltage angles; whereas the ACOPF follows the purest formulation, which comes at a cost [**chatzivasileiadis2018optimization**]. Relaxations are commonly employed to convexify the problem [**ergun2019optimal**], which makes it easier to obtain a satisfactory feasible solution.

Nonetheless, in this document we abstain ourselves from any oversimplification that deviates from the original problem, and we build an Interior Point Method (IPM) to achieve maximum performance as well as avoiding third-party dependencies such as IPOPT.

As explained earlier, the aim of the project is to solve the power flow without simplification of any equation that can give unfeasible results, and that is why the implementation of the solver is done using an IPM. In this chapter, the modelling of the AC-OPF problem as an IPM type of problem is presented. The resulting model will allow the use of this solver for any case study formulated using GridCal. Being a non-convex optimization, the convergence is not guaranteed, although there will be some features that will ease the convergence for grids that have some difficult bounds to satisfy.

5.1 Decision variables

The power flow is modelled using polar form for voltages. The first group of variables correspond to the traditional power flow variables. Then, the bound slack variables are introduced for the line flow limits and the voltage limits. The transformer variables are added next, and finally the DC links $from$ powers. The resulting variable vector is:

$$\boldsymbol{x} = [\boldsymbol{\theta}^T, \boldsymbol{\mathcal{V}}^T, \boldsymbol{P_g}^T, \boldsymbol{Q_g}^T, \boldsymbol{sl_{sf}}^T, \boldsymbol{sl_{st}}^T, \boldsymbol{sl_{vmax}}^T, \boldsymbol{sl_{vmin}}^T, \boldsymbol{m_p}^T, \boldsymbol{\tau}^T, \boldsymbol{P_{dc}}^T]^T \tag{27}$$

where $\boldsymbol{\theta}$ is the vector of size N of voltage angles, $\boldsymbol{\mathcal{V}}$ is the vector of size N of voltage magnitudes, $\boldsymbol{P_g}$ and $\boldsymbol{Q_g}$ are the vectors of size Ng of active and reactive power generation, $\boldsymbol{sl_{sf}}$ and $\boldsymbol{sl_{st}}$ are the vectors of size nll of slack variables for the line flow limits, $\boldsymbol{sl_vmax}$ and $\boldsymbol{sl_{vmin}}$ are the vectors of size npq of slack variables for the voltage limits, $\boldsymbol{tapm}$ and $\boldsymbol{tapt}$ are the vectors of transformer tap positions, sized $ntapm$ and $ntapt$ respectively, and $\boldsymbol{Pfdc}$ is the vector of DC

link $from$ powers, sized ndc.

In total, there are $2N + 2Ng + 2nll + 2npq + ntapm + ntapt + ndc$ decision variables.

5.2 Objective function

Firstly, the objective function is described as the minimization of cost of operation of the grid. This cost is modelled as a quadratic cost with respect to the active power generation, and linear with respect to the limit violation slacks. This means that the generation has to be as cheap as the limitations allow, while this limitation can be surpassed at a greater cost in order to be able to solve complex grid states. For instance, a grid with a high demand state can have some lines slightly overloaded for short periods of time, and exchanging this overload for a penalty can be preferred to not being able to solve the grid state. Of course, the cost has to be such that the overloads are not extreme or common.

The resulting cost function is:

$$\begin{aligned} f(\boldsymbol{x}) = &\sum_{i \in \text{ig}} c_{i_0} + c_{i_1} P_{g_i} + c_{i_2} P_{g_i}^2 \\ &+ \sum_{k \in \text{il}} c_{s_k} (sl_{sf_k} + sl_{st_k}) + \sum_{i \in \text{pq}} c_{sl_{v\,i}} (sl_{vmax_i} + sl_{vmin_i}) \end{aligned} \tag{28}$$

where c_{i_0}, c_{i_1} and c_{i_2} are the coefficients of the quadratic cost function for the active power generation, c_{s_k} is the cost of the slack variables for the line flow limits, $c_{sl_{v_i}}$ is the cost of the slack variables for the voltage limits and ig, il and pq are the sets of dispatchable generators, monitored lines and PQ buses respectively.

5.2.1 Objective function gradient

The gradient of the objective function results in the following vector:

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} \mathbf{0_N} \\ \mathbf{0_N} \\ \boldsymbol{c_1} + 2\boldsymbol{c_2}\boldsymbol{P_g} \\ \mathbf{0_{Ng}} \\ \boldsymbol{c_s} \\ \boldsymbol{c_s} \\ \boldsymbol{c_{sl_v}} \\ \boldsymbol{c_{sl_v}} \\ \mathbf{0_{ntapm}} \\ \mathbf{0_{ntapt}} \\ \mathbf{0_{ndc}} \end{bmatrix} \tag{29}$$

where the subindex $\mathbf{0_n}$ indicates a vector of zeros of size n.

5.2.2 Objective function hessian

The hessian of the objective function is a diagonal matrix, with the following structure:

$$\nabla^2 f(\boldsymbol{x}) = \begin{bmatrix} 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 2[\boldsymbol{c_2}] & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 \end{bmatrix} \tag{30}$$

with only the diagonal elements corresponding to the active power generation variables are non-zero.

5.3 Equality constraints

The vector of equality constraints G for the AC-OPF problem has the following structure:

$$\boldsymbol{G}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{G_P^S} \\ \boldsymbol{G_Q^S} \\ \boldsymbol{G_{slack}^{Th}} \\ \boldsymbol{G_{pv}^V} \end{bmatrix} \tag{31}$$

where $\boldsymbol{G_P^S}$ and $\boldsymbol{G_Q^S}$ are the active and reactive nodal power balances, $\boldsymbol{G_{slack}^{Th}}$ is the slack phase reference, and $\boldsymbol{G_{pv}^V}$ are the voltage equalities for pv buses.

The power balance is calculated in complex form as in 7and then it is split into active and reactive power balances, since the values have to be real numbers for the solver.

To obtain the jacobian and hessian matrices, the G vector needs to be derived twice, including the associated multiplier in the second derivative term. The derivatives are obtained following the Matpower documentation [**zimmerman2011matpower**], directly using the derivatives with respect to the voltage and power variables, and developing with a similar philosophy the derivatives with respect to the additional variables that are introduced in the model.

5.3.1 First derivatives of G

The vector of partial derivatives of the equality constraints G_X^S is:

$$G_X^S = \frac{\partial G^s}{\partial X} = \begin{bmatrix} G_\theta^S & G_V^S & G_P^S & G_Q^S & 0_{n\ x\ sl} & G_{m_p}^S & G_\tau^S & G_{P_{dc}}^S \end{bmatrix} \tag{32}$$

The first derivatives of the power balance equations are obtained form Matpower [**zimmerman2011matpow** as follows:

$$\begin{aligned}
G_\theta^S &= \frac{\partial \boldsymbol{S_{bus}}}{\partial \boldsymbol{\theta}} = [\boldsymbol{I_{bus}^*}]\frac{\partial \boldsymbol{V}}{\partial \boldsymbol{\theta}} + [\boldsymbol{V}]\frac{\partial \boldsymbol{I_{bus}^*}}{\partial \boldsymbol{\theta}} \\
&= \left[\boldsymbol{I_{bus}^*}\right] j[\boldsymbol{V}] + \left[(j\boldsymbol{V})\boldsymbol{I_{bus}^*}\right] \\
&= j[\boldsymbol{V}]\left([\boldsymbol{I_{bus}^*}] - Y_{bus}^*[\boldsymbol{V^*}]\right) \\
G_\mathcal{V}^S &= \frac{\partial \boldsymbol{S_{bus}}}{\partial \boldsymbol{\mathcal{V}}} = \left[\boldsymbol{I_{bus}^*}\frac{\partial \boldsymbol{V}}{\partial \boldsymbol{\mathcal{V}}} + [\boldsymbol{V}]\frac{\partial \boldsymbol{I_{bus}^*}}{\partial \boldsymbol{\mathcal{V}}}\right] \\
&= \left[\boldsymbol{I_{bus}^*}\right][\boldsymbol{E}] + [\boldsymbol{V}]Y_{bus}^*[\boldsymbol{E^*}] \\
&= [\boldsymbol{V}]([\boldsymbol{I_{bus}^*}] + Y_{bus}^*[\boldsymbol{V^*}])[\boldsymbol{\mathcal{V}}]^{-1} \\
G_{P_g}^S &= -C_g \\
G_{Q_g}^S &= -jC_g
\end{aligned} \tag{33}$$

Where $[\boldsymbol{E}] = [\boldsymbol{V}]\boldsymbol{\mathcal{V}}^{-1}$. The derivatives with respect to the tap variables have been derived with the following expressions, starting from the admittance primitives which are the quantities that depend on the tap variables as seen in Equations 1 and 2. These primitives are used to calculate the $from$ and to power flowing through a line and its primitives with the following expression:

$$
\begin{aligned}
\boldsymbol{S_f} &= \boldsymbol{V_f} \boldsymbol{I_f^*} \\
\boldsymbol{S_t} &= \boldsymbol{V_t} \boldsymbol{I_t^*} \\
\boldsymbol{I_f} &= y_{ff} \boldsymbol{V_f} + y_{ft} \boldsymbol{V_t} \\
\boldsymbol{I_t} &= y_{tf} \boldsymbol{V_f} + y_{tt} \boldsymbol{V_t}
\end{aligned}
\tag{34}
$$

To obtain the power injection per bus, we compose the S vector using the connectivity matrices as follows:

$$
\boldsymbol{S_{bus}} = C_f^T \boldsymbol{S_f} + C_t^T \boldsymbol{S_t} \tag{35}
$$

The equality constraints for the power flow that have to be derived are the ones described by Equation 7. Deriving the expressions written above for the primitive admittances, we obtain the following expressions:

$$
\begin{aligned}
\frac{\partial \boldsymbol{G^S}}{\partial \boldsymbol{m_p}} &= \frac{\partial \boldsymbol{S_{bus}}}{\partial \boldsymbol{m_p}} = Cf^T \frac{\partial \boldsymbol{S_f}}{\partial \boldsymbol{m_p}} + Ct^T \frac{\partial \boldsymbol{S_t}}{\partial \boldsymbol{m_p}} \\
\frac{\partial \boldsymbol{G^S}}{\partial \boldsymbol{\tau}} &= \frac{\partial \boldsymbol{S_{bus}}}{\partial \boldsymbol{\tau}} = Cf^T \frac{\partial \boldsymbol{S_f}}{\partial \boldsymbol{\tau}} + Ct^T \frac{\partial \boldsymbol{S_t}}{\partial \boldsymbol{\tau}} \\
\frac{\partial S_{f_k}}{\partial m_{p_i}} &= V_{f_k} \left(\frac{-2(y_{s_k} V_{f_k})^*}{m_{p_i}^3} + \frac{(y_{s_k} V_{t_k})^*}{m_{p_i}^2 e^{j\tau}} \right) \\
\frac{\partial S_{t_k}}{\partial m_{p_i}} &= V_{t_k} \frac{(y_{s_k} V_{f_k})^*}{m_{p_i}^2 e^{-j\tau_i}} \\
\frac{\partial S_{f_k}}{\partial \tau_i} &= V_{f_k} \frac{j(y_{s_k} V_{t_k})^*}{m_{p_i} e^{j\tau_i}} \\
\frac{\partial S_{t_k}}{\partial \tau_i} &= V_{t_k} \frac{-j(y_{s_k} V_{f_k})^*}{m_{p_i} e^{-j\tau_i}}
\end{aligned}
\tag{36}
$$

Each of the individual S_{f_k} and S_{t_k} derivative with respect to each i transformer variable will be stored in the position of their corresponding matrix as $\frac{dS_f}{dm_p}_{ki}$ and $\frac{dS_f}{dm_p}_{ki}$, with size L x $ntapm$ and L x $ntapt$ respectively.

The derivative of $G^S_{P_{dc}}$ can be obtained using the indexing of the buses participant of a DC link, which are listed in the lists fdc and tdc. Each link DC is also indexed in the list dc, of length ndc. The derivative is obtained as follows:

$$\begin{cases} G^S_{P_{dc}}\{fdc, link\} = 1 \quad \forall link = (fdc, tdc) \in dc \\ G^S_{P_{dc}}\{tdc, link\} = -1 \quad \forall link = (fdc, tdc) \in dc \\ 0 \quad \text{otherwise} \end{cases} \tag{37}$$

The complete matrix G_X can be obtained concatenating the following submatrices:

$$G_X = \begin{bmatrix} \boldsymbol{\mathcal{R}(G_X^{S^\top})} \\ \boldsymbol{\mathcal{I}(G_X^{S^\top})} \\ \boldsymbol{G_X^{Th^\top}} \\ \boldsymbol{G_X^{\mathcal{V}^\top}} \end{bmatrix} \tag{38}$$

where G^{Th} and $G^{\mathcal{V}}$ are the slack phase and voltage magnitude equalities, and their derivatives have 1 in the fixed buses positions and 0 otherwise.

5.3.2 Second derivatives of G

$$\begin{bmatrix} G_{\theta\theta} & G_{\theta\mathcal{V}} & G_{\theta P_g} & G_{\theta Q_g} & G_{\theta sl} & G_{\theta m_p} & G_{\theta\tau} & G_{\theta P_{dc}} \\ G_{\mathcal{V}\theta} & G_{\mathcal{V}\mathcal{V}} & G_{\mathcal{V}P_g} & G_{\mathcal{V}Q_g} & G_{\mathcal{V}sl} & G_{\mathcal{V}m_p} & G_{\mathcal{V}\tau} & G_{\mathcal{V}P_{dc}} \\ G_{P_g\theta} & G_{P_g\mathcal{V}} & G_{P_gP_g} & G_{P_gQ_g} & G_{P_gsl} & G_{P_gm_p} & G_{P_g\tau} & G_{P_gP_{dc}} \\ G_{Q_g\theta} & G_{Q_g\mathcal{V}} & G_{Q_gP_g} & G_{Q_gQ_g} & G_{Q_gsl} & G_{Q_gm_p} & G_{Q_g\tau} & G_{Q_gP_{dc}} \\ G_{sl\theta} & G_{sl\mathcal{V}} & G_{slP_g} & G_{slQ_g} & G_{slsl} & G_{slm_p} & G_{sl\tau} & G_{slP_{dc}} \\ G_{m_p\theta} & G_{m_p\mathcal{V}} & G_{m_pP_g} & G_{m_pQ_g} & G_{m_psl} & G_{m_pm_p} & G_{m_p\tau} & G_{m_pP_{dc}} \\ G_{\tau\theta} & G_{\tau\mathcal{V}} & G_{\tau P_g} & G_{\tau Q_g} & G_{\tau sl} & G_{\tau m_p} & G_{\tau\tau} & G_{\tau P_{dc}} \\ G_{P_{dc}\theta} & G_{P_{dc}\mathcal{V}} & G_{P_{dc}P_g} & G_{P_{dc}Q_g} & G_{P_{dc}sl} & G_{P_{dc}m_p} & G_{P_{dc}\tau} & G_{P_{dc}P_{dc}} \end{bmatrix} \tag{39}$$

The first set of second derivatives in the upper left (Power Flow variables $((\theta, \mathcal{V}, P_g, Q_g))$, noted $\boldsymbol{X_{PF}}$) is obtained from the Technical Note 2 of Matpower [**zimmerman2011matpower**].

$$
\begin{aligned}
G^S_{XX_{PF}}(\boldsymbol{\lambda}) &= \frac{\partial}{\partial \boldsymbol{X_{PF}}}(\boldsymbol{G^S}^{\top}_{\boldsymbol{X_{PF}}}\boldsymbol{\lambda}) \\
&= \begin{bmatrix} G^S_{\boldsymbol{\theta\theta}}(\boldsymbol{\lambda}) & G^S_{\boldsymbol{\theta}V}(\boldsymbol{\lambda}) & 0 & 0 \\ G^S_{V\boldsymbol{\theta}}(\boldsymbol{\lambda}) & G^S_{VV}(\boldsymbol{\lambda}) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\
G^S_{\boldsymbol{\theta\theta}}(\boldsymbol{\lambda}) &= \frac{\partial}{\partial \boldsymbol{\theta}}({G^S}^{\top}_{\boldsymbol{\theta}}\boldsymbol{\lambda}) \\
&= \left[\boldsymbol{V^*}\right](Y^{*\top}_{bus}\left[\boldsymbol{V}\right]\left[\boldsymbol{\lambda}\right] - \left[Y^{*\top}_{bus}[\boldsymbol{V}]\boldsymbol{\lambda}\right]) \\
&\quad + \left[\boldsymbol{\lambda}\right]\left[\boldsymbol{V}\right]\left(Y^*_{bus}\left[\boldsymbol{V^*}\right] - \left[\boldsymbol{I^*_{bus}}\right]\right) \\
G^S_{\boldsymbol{\mathcal{V}\theta}}(\boldsymbol{\lambda}) &= \frac{\partial}{\partial \boldsymbol{\theta}}({G^S}^{\top}_{\boldsymbol{\mathcal{V}}}\boldsymbol{\lambda}) \\
&= j[\boldsymbol{\mathcal{V}}]^{-1}([\boldsymbol{V^*}](Y^*_{bus}[\boldsymbol{V}][\boldsymbol{\lambda}] - [Y^{*\top}_{bus}[\boldsymbol{V}]\boldsymbol{\lambda}]) \\
&\quad - [\boldsymbol{\lambda}][\boldsymbol{V}](Y^*_{bus}[\boldsymbol{V^*}] - [\boldsymbol{I^*_{bus}}])) \\
G^S_{\boldsymbol{\mathcal{V}\mathcal{V}}}(\boldsymbol{\lambda}) &= \frac{\partial}{\partial \boldsymbol{\mathcal{V}}}({G^S}^{\top}_{\boldsymbol{\mathcal{V}}}\boldsymbol{\lambda}) \\
&= [\boldsymbol{\mathcal{V}}]^{-1}\left([\boldsymbol{\lambda}][\boldsymbol{V}]Y^*_{bus}[\boldsymbol{V^*}] + [\boldsymbol{V^*}]Y^*_{bus}[\boldsymbol{V}][\boldsymbol{\lambda}]\right)[\boldsymbol{\mathcal{V}}]^{-1}
\end{aligned}
\tag{40}
$$

The rest of the second derivatives have been obtained following a similar strategy to add the multipliers. The derivation starts again with the $from$ and to expressions, and is as follows:

$$
\begin{aligned}
\frac{\partial^2 S_{f_k}}{\partial m_{p_i}\partial m_{p_i}} &= V_{f_k}\left(\frac{6(y_{s_k}V_{f_k})^*}{m_{p_i}^4} - \frac{2(y_{s_k}V_{t_k})^*}{m_{p_i}^3 e^{j\tau_i}}\right)\\
\frac{\partial^2 S_{t_k}}{\partial m_{p_i}\partial m_{p_i}} &= V_{t_k}\frac{-2(y_{s_k}V_{f_k})^*}{m_{p_i}^3 e^{-j\tau_i}}\\
\frac{\partial^2 S_{f_k}}{\partial \tau_i\partial \tau_i} &= V_{f_k}\frac{(y_{s_k}V_{t_k})^*}{m_{p_i} e^{j\tau_i}}\\
\frac{\partial^2 S_{t_k}}{\partial \tau_i\partial \tau_i} &= V_{t_k}\frac{(y_{s_k}V_{f_k})^*}{m_{p_i} e^{-j\tau_i}}\\
\frac{\partial^2 S_{f_k}}{\partial m_{p_i}\partial \tau_i} &= V_{f_k}\frac{-j(y_{s_k}V_{t_k})^*}{m_{p_i}^2 e^{j\tau_i}}\\
\frac{\partial^2 S_{t_k}}{\partial m_{p_i}\partial \tau_i} &= V_{t_k}\frac{j(y_{s_k}V_{f_k})^*}{m_{p_i}^2 e^{-j\tau_i}}\\
\frac{\partial S_{f_k}}{\partial m_{p_i}\partial v_{f_k}} &= \frac{V_{f_k}}{v_{f_k}}\left(\frac{-4(y_{s_k}V_{f_k})^*}{m_{p_i}^3} + \frac{(y_{s_k}V_{t_k})^*}{m_{p_i}^2 e^{j\tau_i}}\right)\\
\frac{\partial S_{t_k}}{\partial m_{p_i}\partial v_{f_k}} &= \frac{V_{t_k}}{v_{f_k}}\frac{(y_{s_k}V_{f_k})^*}{m_{p_i}^2 e^{-j\tau_i}}\\
\frac{\partial S_{f_k}}{\partial m_{p_i}\partial v_{t_k}} &= \frac{V_{f_k}}{v_{f_t}}\left(\frac{-2(y_{s_k}V_{f_k})^*}{m_{p_i}^3} + \frac{(y_{s_k}V_{t_k})^*}{m_{p_i}^2 e^{j\tau_i}}\right)\\
\frac{\partial S_{t_k}}{\partial m_{p_i}\partial v_{t_k}} &= \frac{V_{t_k}}{v_{f_t}}\frac{(y_{s_k}V_{f_k})^*}{m_{p_i}^2 e^{-j\tau_i}}\\
\frac{\partial S_{f_k}}{\partial \tau_i\partial v_{f_k}} &= \frac{V_{f_k}}{v_{f_k}}\frac{j(y_{s_k}V_{t_k})^*}{m_{p_i} e^{j\tau_i}}\\
\frac{\partial S_{t_k}}{\partial \tau_i\partial v_{f_k}} &= \frac{V_{t_k}}{v_{f_k}}\frac{-j(y_{s_k}V_{f_k})^*}{m_{p_i} e^{-j\tau_i}}\\
\frac{\partial S_{f_k}}{\partial \tau_i\partial v_{f_k}} &= \frac{V_{f_k}}{v_{f_t}}\frac{j(y_{s_k}V_{t_k})^*}{m_{p_i} e^{j\tau_i}}\\
\frac{\partial S_{t_k}}{\partial \tau_i\partial v_{f_k}} &= \frac{V_{t_k}}{v_{f_t}}\frac{-j(y_{s_k}V_{f_k})^*}{m_{p_i} e^{-j\tau_i}}\\
\frac{\partial S_{f_k}}{\partial m_{p_i}\partial \theta_{f_k}} &= V_{f_k}\frac{j(y_{s_k}V_{t_k})^*}{m_{p_i}^2 e^{j\tau_i}}\\
\frac{\partial S_{t_k}}{\partial m_{p_i}\partial \theta_{f_k}} &= V_{t_k}\frac{-j(y_{s_k}V_{f_k})^*}{m_{p_i}^2 e^{-j\tau_i}}\\
\frac{\partial S_{f_k}}{\partial m_{p_i}\partial \theta_{t_k}} &= V_{f_k}\frac{-j(y_{s_k}V_{t_k})^*}{m_{p_i}^2 e^{j\tau_i}}\\
\frac{\partial S_{t_k}}{\partial m_{p_i}\partial \theta_{t_k}} &= V_{t_k}\frac{j(y_{s_k}V_{f_k})^*}{m_{p_i}^2 e^{-j\tau_i}}\\
\frac{\partial S_{f_k}}{\partial \tau_i\partial \theta_{f_k}} &= V_{f_k}\frac{-(y_{s_k}V_{t_k})^*}{m_{p_i} e^{j\tau_i}}\\
\frac{\partial S_{t_k}}{\partial \tau_i\partial \theta_{f_k}} &= V_{t_k}\frac{-(y_{s_k}V_{f_k})^*}{m_{p_i} e^{-j\tau_i}}\\
\frac{\partial S_{f_k}}{\partial \tau_i\partial \theta_{f_k}} &= V_{f_k}\frac{(y_{s_k}V_{t_k})^*}{m_{p_i} e^{j\tau_i}}\\
\frac{\partial S_{t_k}}{\partial \tau_i\partial \theta_{f_k}} &= V_{t_k}\frac{(y_{s_k}V_{f_k})^*}{m_{p_i} e^{-j\tau_i}}
\end{aligned}
\tag{41}
$$

We have to consider the crossed partial derivative $\frac{\partial^2}{\partial m_{p_i} \partial \tau_i}$ for those lines with transformer that controls both variables. To compose the hessian matrix, we perform as the following example for all the crossed derivatives:

$$\begin{aligned} \frac{dS_{bus}}{dmdm}_{ii} &= \Re\left(\frac{\partial^2 S_{f_k}}{\partial m_{p_i} \partial m_{p_i}} \lambda_f + \frac{\partial^2 S_{t_k}}{\partial m_{p_i} \partial m_{p_i}} \lambda_t\right) \\ &+ \Im\left(\frac{\partial^2 S_{f_k}}{\partial m_{p_i} \partial m_{p_i}} \lambda_{f+N} + \frac{\partial^2 S_{t_k}}{\partial m_{p_i} \partial m_{p_i}} \lambda_{t+N}\right) \end{aligned} \tag{42}$$

Since we have to take into account for the active and reactive power constraints, we have to divide the expression into real and imaginary and then use the corresponding multiplier. We compose the resulting hessian by concatenating the complete matrix as in the previous case.

5.4 Inequality constraints

We proceed similarly to the equality constraints. The vector of inequality constraints H for the AC-OPF problem has the following structure:

$$\boldsymbol{H}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{H^{S_f}} \\ \boldsymbol{H^{S_t}} \\ \boldsymbol{H^{V_u}} \\ \boldsymbol{H^{P_u}} \\ \boldsymbol{H^{Q_u}} \\ \boldsymbol{H^{V_l}} \\ \boldsymbol{H^{P_l}} \\ \boldsymbol{H^{Q_l}} \\ \boldsymbol{H^{sl}} \\ \boldsymbol{H^{m_{p_u}}} \\ \boldsymbol{H^{m_{p_l}}} \\ \boldsymbol{H^{\tau_u}} \\ \boldsymbol{H^{\tau_l}} \\ \boldsymbol{H^{Q_{max}}} \\ \boldsymbol{H^{Pf_{dc_u}}} \\ \boldsymbol{H^{Pf_{dc_l}}} \end{bmatrix} \tag{43}$$

The first derivatives of the power flows are the most complex ones to get, since they will be derived from the $from$ and to powers, and the constraints has its value squared.

$$\boldsymbol{H_f^S} = [\boldsymbol{S_f^*}]\boldsymbol{S_f} - \boldsymbol{S_{max}^2} \tag{44}$$

$$\begin{aligned} H_X^f &= 2(\Re([\boldsymbol{S_f}])\Re(S_X^f) + \Im([\boldsymbol{S_f}])\Im(S_X^f)) \\ H_{XX}^f(\boldsymbol{\mu}) &= 2\Re(S_{XX}^f([\boldsymbol{S_f^*}]\boldsymbol{\mu}) + {S_X^f}^T[\boldsymbol{\mu}]S_X^f) \end{aligned} \tag{45}$$

And similarly for the H^t $from$ constraints. The derivatives with respect of the PF variables are obtained from the Matpower documentation [**zimmerman2011matpower**] again:

$$\begin{aligned} S_{\boldsymbol{\theta}}^f &= \left[\boldsymbol{I_f^*}\right]\frac{\partial \boldsymbol{V_f}}{\partial \boldsymbol{\theta}} + \left[\boldsymbol{V_f}\right]\frac{\partial \boldsymbol{I_f^*}}{\partial \boldsymbol{\theta}} \\ &= \left[\boldsymbol{I_f^*}\right] jC_f[\boldsymbol{V}] + [C_f\boldsymbol{V}](j\bar{Y_f}[\boldsymbol{V}])^* \\ &= j\left([\boldsymbol{I_f^*}]C_f[\boldsymbol{V}] - [C_f\boldsymbol{V}]Y_f^*[\boldsymbol{V^*}]\right) \\ S_{\boldsymbol{\mathcal{V}}}^f &= \left[\boldsymbol{I_f^*}\right]\frac{\partial \boldsymbol{V_f}}{\partial \boldsymbol{\mathcal{V}}} + \left[\boldsymbol{V_f}\right]\frac{\partial \boldsymbol{I_f^*}}{\partial \boldsymbol{\mathcal{V}}} \\ &= \left[\boldsymbol{I_f^*}\right] C_f[\boldsymbol{E}] + [C_f\boldsymbol{V}]Y_f^*[\boldsymbol{E^*}] \\ S_{\boldsymbol{P_g}}^f &= 0 \\ S_{\boldsymbol{Q_g}}^f &= 0 \end{aligned} \tag{46}$$

$$\begin{aligned} S_{\boldsymbol{\theta\theta}}^f(\boldsymbol{\mu}) &= \frac{\partial}{\partial \boldsymbol{\theta}}\left(S_{\boldsymbol{\theta}}^f\right)^T\boldsymbol{\mu} \\ &= [\boldsymbol{V^*}]{Y_f^*}^\top[\boldsymbol{\mu}]C_f[\boldsymbol{V}] + [\boldsymbol{V}]C_f^\top[\boldsymbol{\mu}]Y_f^*[\boldsymbol{V^*}] \\ &- [{Y_f^*}^\top[\boldsymbol{\mu}]C_f\boldsymbol{V}][\boldsymbol{V^*}] - [C_f^T[\boldsymbol{\mu}]Y_f^*\boldsymbol{V^*}][\boldsymbol{V}] \\ S_{\boldsymbol{\mathcal{V}\theta}}^f(\boldsymbol{\mu}) &= \frac{\partial}{\partial \boldsymbol{\theta}}\left(S_{\boldsymbol{\mathcal{V}}}^f\right)^\top\boldsymbol{\mu} \\ &= j[\boldsymbol{\mathcal{V}}]^{-1}([\mathbf{V^*}]{Y_f^*}^\top[\boldsymbol{\mu}]C_f[\mathbf{V}] - [\mathbf{V}]C_f^\top[\boldsymbol{\mu}]Y_f^*[\mathbf{V^*}] \\ &- [{Y_f^*}^\top[\boldsymbol{\mu}]C_f\mathbf{V}][\mathbf{V^*}] + [C_f^\top[\boldsymbol{\mu}]Y_f^*\mathbf{V^*}][\mathbf{V}]) \\ S_{\boldsymbol{\mathcal{V}\mathcal{V}}}^f(\boldsymbol{\mu}) &= \frac{\partial}{\partial \boldsymbol{\mathcal{V}}}\left(S_{\boldsymbol{\mathcal{V}}}^f\right)^\top\boldsymbol{\mu} \\ &= [\boldsymbol{\mathcal{V}}]^{-1}\left([\mathbf{V^*}]{Y_f^*}^\top[\boldsymbol{\mu}]C_f[\mathbf{V}] + [\mathbf{V}]C_f^\top[\boldsymbol{\mu}]Y_f^*[\mathbf{V^*}]\right)[\boldsymbol{\mathcal{V}}]^{-1} \end{aligned} \tag{47}$$

The derivatives with respect to the tap variables have been derived using $\boldsymbol{H}$, the expressions for $\boldsymbol{S_f}$, $\boldsymbol{S_t}$ and its derivatives obtained in the power balance derivatives in Equations 36.

6 Results

After implementing the software in the GridCal package, now it is time to test how does this solver perform when compared to already available solutions. The benchmark used for this comparison will be the Matpower solver, an open-source package that has been used in many works as a reference for the AC-OPF problem. The objective is to have a similar runtime and number of iterations for the base AC-OPF problem (the model already used by Matpower's solver), and to see how well does it scale when the new features are added.

The grids used for this benchmark are a combination of small grids with some interesting configurations that allow some testing of the new features, and larger grids that will be used to see if the solver is capable to solve them and what is the computational cost of doing so. The largest grid include the Great Britain national grid (an adaptation), which can give some insight on how the solver performs in a real-world scenario.

The results obtained have been obtained in a computer with the following specifications:

- CPU: Intel Core i7-1165G7 @ 2.80GHz
- RAM: 16GB DDR4
- GPU: Intel Iris Xe Graphics
- OS: Windows 11
- IDE: PyCharm
- Python version: 3.10
- Matpower version: 5.1.16 (Python adaptation)

And the tolerance requested for the convergence conditions is 10^{-6}.

There are two possible initializations for the solver, which are the following:

- **Power flow initialization:** The solver uses GridCal's built-in power flow with the default values introduced in the grid model as the initial values for the variables. The results of the power flow solver are used as the starting point for the optimization variables
- **Basic initialization:** This is the initialization used by Matpower, which sets the variables at the center of the domain. Although it seems simpler, for some cases it could be better to ensure all the variables are inside their domain.

The use of bound slacks has also been tested for both initialization options, yielding a total of 4

different options per case. The results will be shown in the following subsections.

6.1 Error evolution

The first comparison carried out will be the error evolution for the different cases. The graphs will portray the error at each iteration for the different initialization settings, and will be compared to the error obtained by Matpower's solver. The error is the maximum value of the expressions 26 at each iteration. This error corresponds to the maximum value of the convergence conditions, which are calculated for each of the Lagrangian derivative, and including the γ value. The error should always tend to decrease, although sometimes it does so really slowly or it even goes up for some iterations. This could mean that the path taken by the interior point method is exceeding some limits, and the resulting new point is worse than the previous. Excessive increases of the error are normally avoided by the step control implemented in the solver, although it does not prevent these increases in problems that are ill-conditioned or even unsolvable.

6.1.1 Case 9-bus

The first case is a 9-bus system which can be useful to see the behavior of the algorithm in a small system, while it can be portrayed with sufficient clarity in this paper using the GridCal GUI.

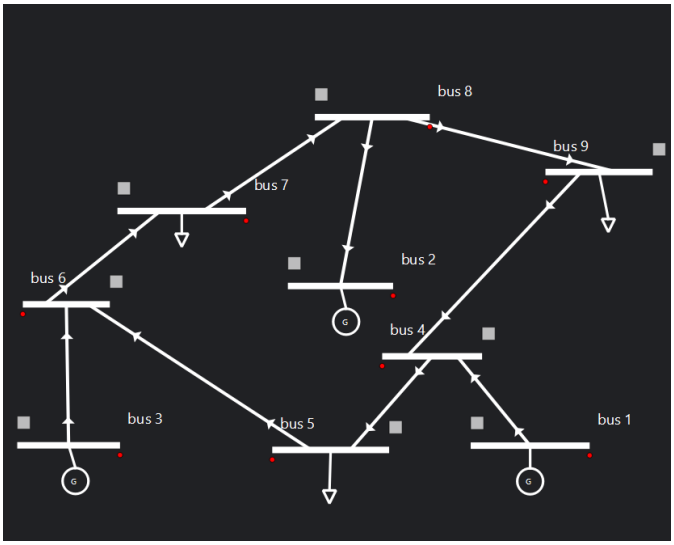

Figure 3: Schematic of the 9-bus grid.

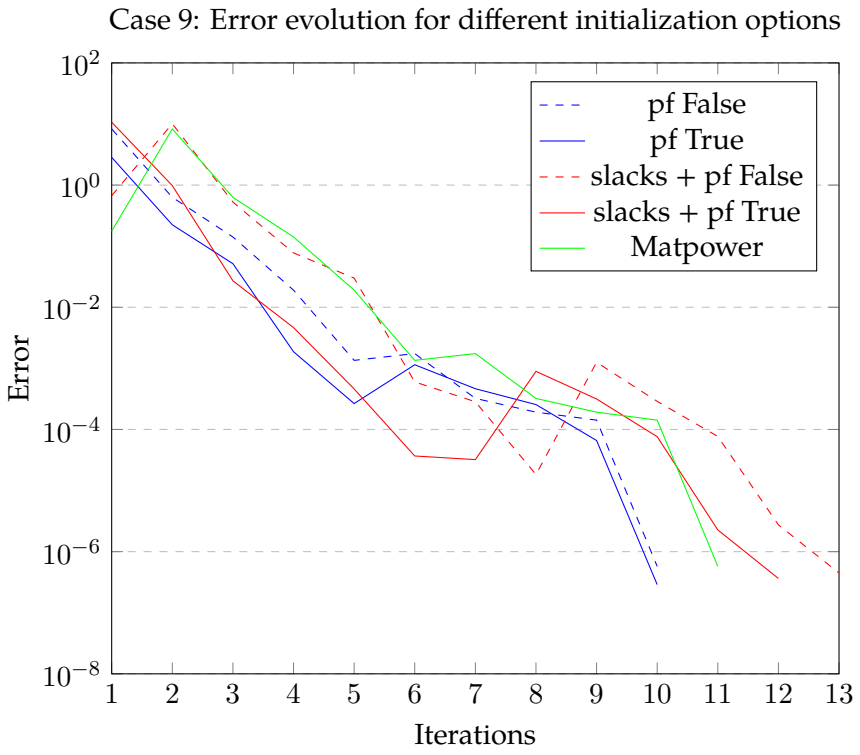

Figure 4: Error evolution for the 9-bus system for different initialization options.

It can be seen in Figure 4 that for such a small grid, there is not a significant difference between the different initialization options. The error evolution is very similar for all of them, and the error obtained is very close to the one obtained by Matpower, being the option without bound slacks slightly better as the vector of variables is considerably smaller. If there are no issues with the grid, the initialization with bound slacks is not necessary for this case.

The result can be visualized in GridCal as seen in Figure 5

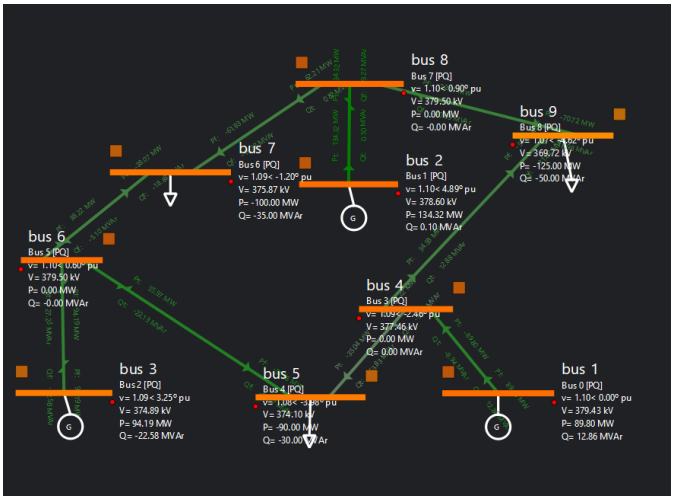

Figure 5: Caption for the solved case 9.

6.1.2 Case Pegase89

The next grid studied is a case with 89 buses obtained from the Pegase database [**pegase89**], with an interesting result shown in Figure 6.

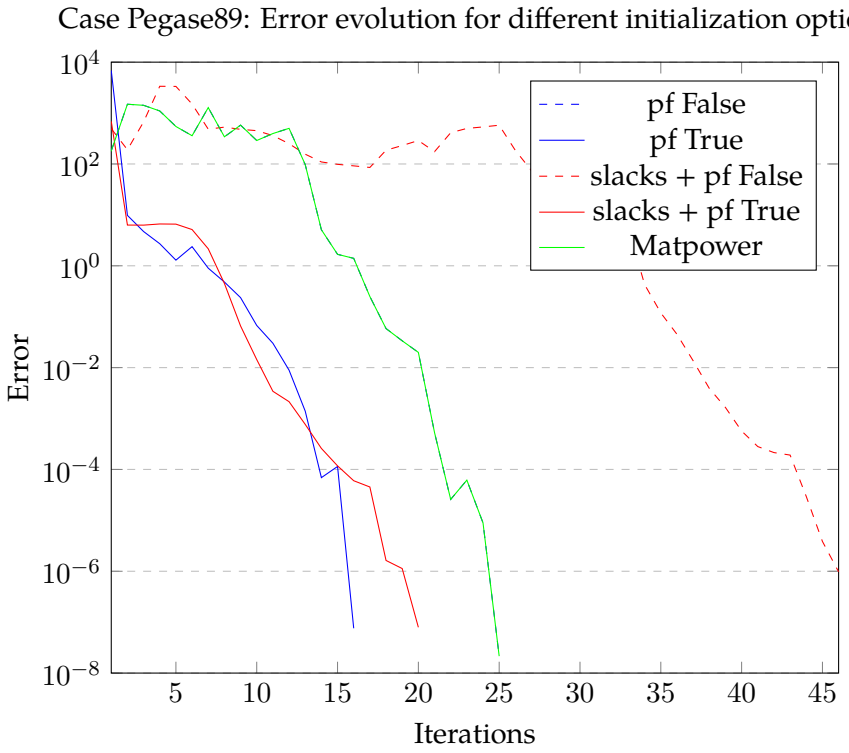

Figure 6: Error evolution for the Pegase89 system for different initialization options.

Firstly, it should be noted that the solution without bound slacks and with the power flow ini-

tialization disabled follows the exact same evolution as the Matpower solver. This did not happen in the previous case, but it seems to be due to the GridCal parser processing of the model. For some cases during the developing of the software, there were cases were unsolvable due to some of its parameters not being directly specified in the model. For instance, some line ratings were not specified, and while the Matpower solver would then ignore some of these limits, the GridCal parser would not be able to process it in a correct way. To solve this, the parser has been adjusted to set a default value in case the lines are monitored but the rating is specified. Many other small nuances during the model loading could be the source of this difference in the convergence.

The other relevant observation that can be done is that the power flow initialization is doing a great job in improving the convergence of the problem. In a real scenario situation, the user would input a grid model that is already in a feasible state, although it may not be optimal, but it might be close to what an optimum generation profile would be. In this case, the OPF using an initialization with power flow takes almost half of the iterations in the case without bound slacks, and a third in the case with bound slacks.

One last observation is that bound slacks are not useful in this case either. The error evolution is very similar to the one without bound slacks, and the number of variables is considerably larger, which will make the problem a bit heavier to run as it will be shown in a later subsection.

6.1.3 Case 300

The next case is a 300-bus system, which is a considerably larger grid than the previous ones. The results are shown in Figure 7.

The performance of the solver developed in this work outclasses the one of Matpower in this case. Considering that the solver is based on the same principles, this means that the GridCal modelled problem is better conditioned than the one modelled by Matpower, which is why the solver has been developed in such an environment.

6.1.4 Case GB

Last case shown is a country sized 2223-bus system that has been provided directly by Redeia, which models the Great Britain grid. The results are shown in Figure 8.

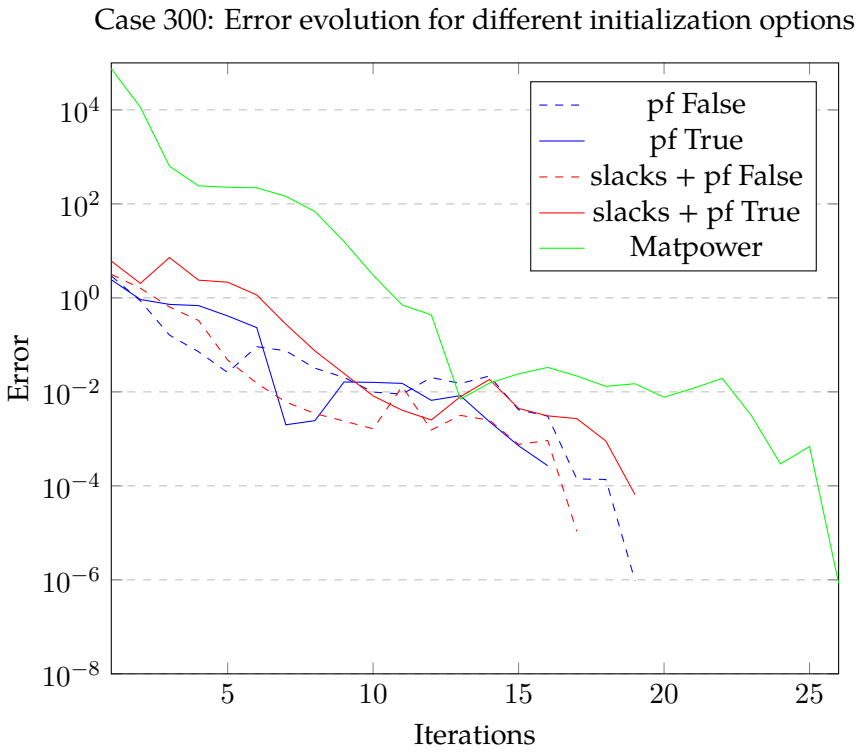

Figure 7: Error evolution for the 300-bus system for different initialization options.

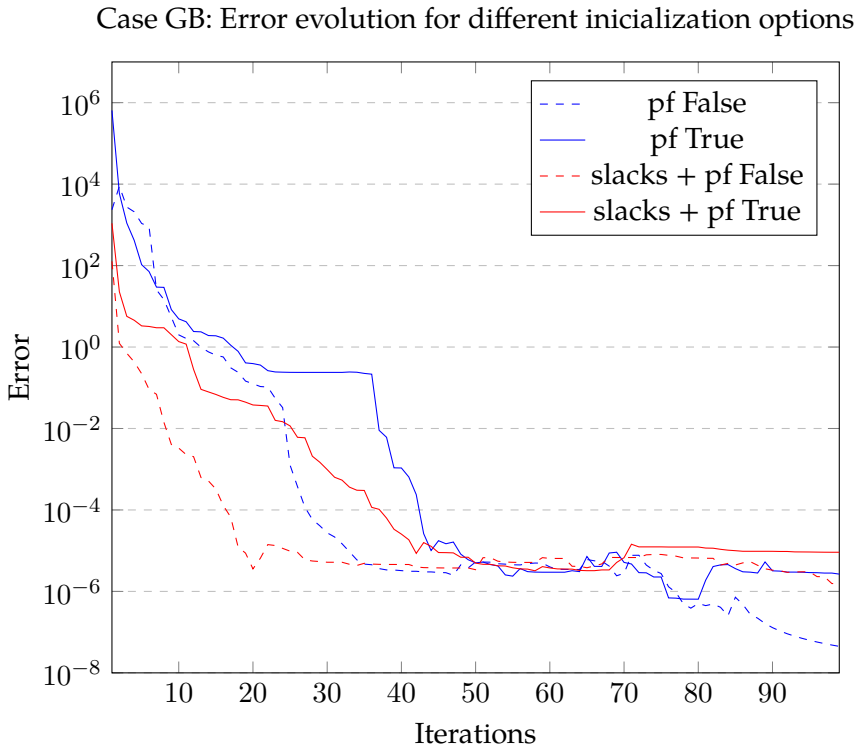

Figure 8: Error evolution for the Great Britain grid for different initialization options.

In this case, there is no comparative with the Matpower solver since the data is stored in a file that can only be managed in GridCal, and the model is not public. The results shows a really promising evolution for a real use case, obtaining a low error in a few iterations. The slacks initialization option can be preferred in this case, which is to be expected since it is a case that has not been tested before and it might correspond to a difficult grid state (meaning the limits could be a bit strict, some generators/lines could disabled or the load profile could be high).

Despite the good performance during the first steps, it stagnates when asking for lower tolerance values, a side effect of adding a lot of variables to the problem.

6.2 Effect of tap variables control - Case IEEE14

Once having seen the performance of the solver in some study cases, it is time to analyze the effect of the tap variables control. This control is a feature that allows the solver to manage the tap variables of the transformers in the grid, which are the ones that can be modified to control the voltage levels in the buses. This control can be done in two ways: by setting the tap variables as fixed variables or by setting them as optimization variables. The first option is the one that has been used in the previous subsections, and the second one is the one that is going to be tested in this one.

The grid used is the IEEE14 model grid, which is a 14-bus grid that includes transformers. Firstly, the error evolution is compared to see the effect of adding this tap variables.

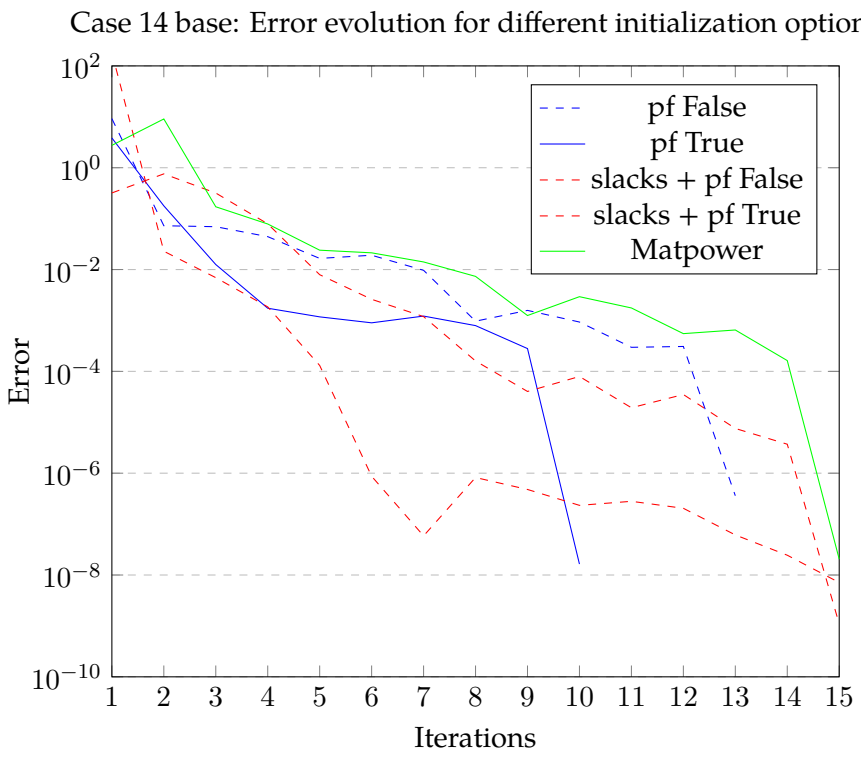

Figure 9: Error evolution for the case 14 for different initialization options.

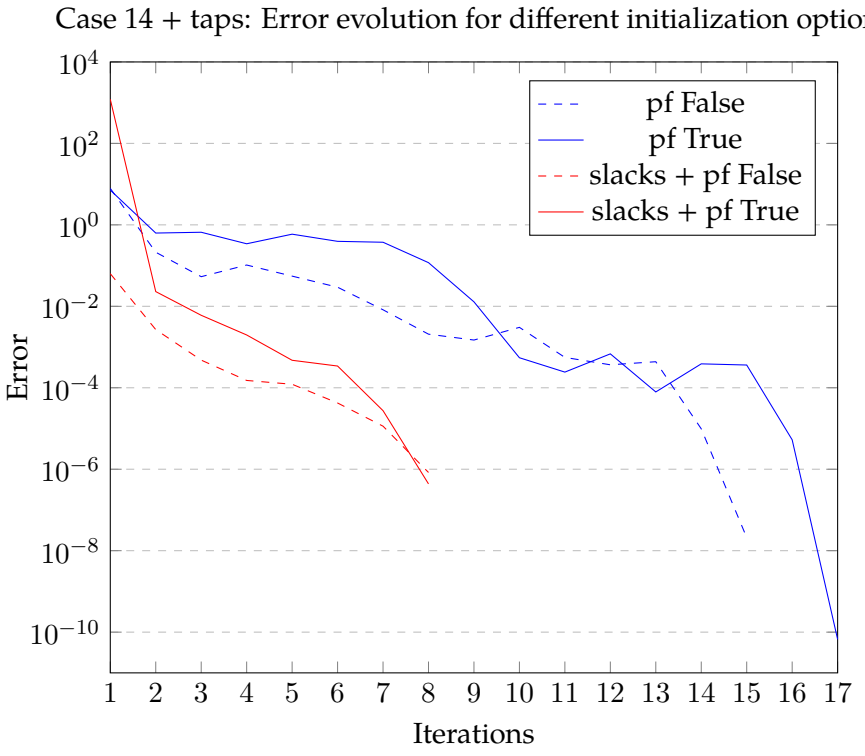

Figure 10: Error evolution for the case 14 with tap variables for different initialization options.

There are two important remarks obtained from these error evolutions. Firstly, the case is solved in fewer iterations in the version of the solver developed in this solver. Secondly, the usage of bound slacks improves convergence for the case with tap variables. Checking now the obtained results, Figures 9 and 10 shows the difference between all the studied cases. For simplicity, the comparison will be made only for the cases with power flow initialization and without bound slacks, as they add to the cost function and the interest of this analysis is testing the effect of using tap optimization.

Figures 11 and 12 show the values of the solution for the bus voltages and generators. Note that there are no differences between the case solved with Matpower and the developed software. The case with controlled tap variable has some slight differences, specially regarding the voltage magnitude. The impact in the generation is also noticeable for the reactive power generation, which has shifted mostly from generator 1 to generator 3.

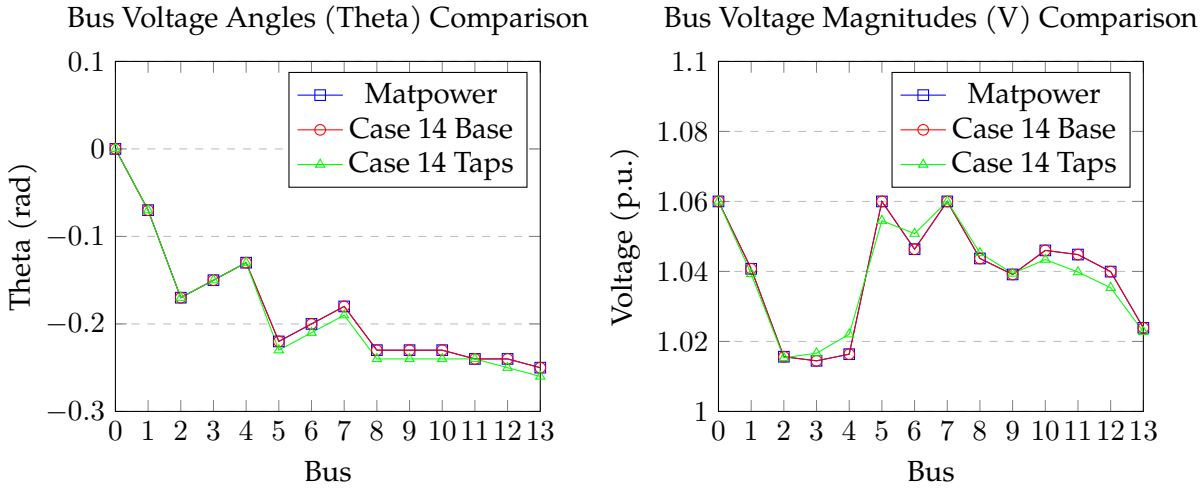

Figure 11: Voltage comparison for the case 14 considering tap variables.

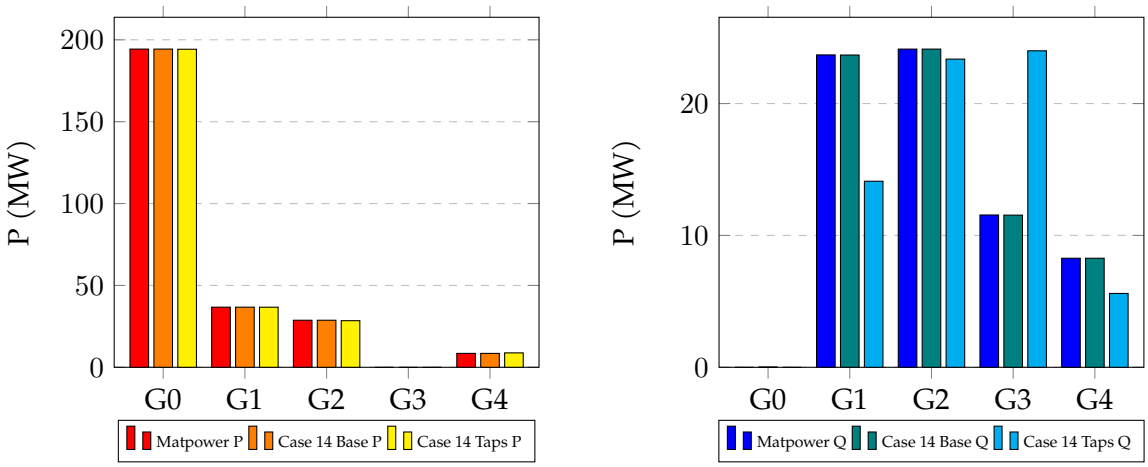

Figure 12: Generation comparison for case 14.

Table 1 shows the setpoints for the tap variables of the transformers in the grid as obtained with the solver developed. Note that the transformers that only have one variable controlled have a N/C as a solution.

Controlled trafos	**m_p (p.u)**	τ **(rad)**
Branch 17	0.965876	0.013175
Branch 18	N/C	0.023833
Branch 19	0.974728	N/C

Table 1: Tap variable setpoints for the transformers of the corresponding branches. N/C stands for Non-Controlled variables

The important result is shown in table 2, which shows the impact of being able to control the tap variables setpoints in the cost of operation. Of course, this impact may not seem big in this case, representing a reduction of cost of only 0.03%. However, this result has only considered a small grid, without checking for real generation costs, and only controlling three transformers. And still, considering the total expenses of a country-sized grid, each small optimization applied to the system can lead to huge savings.

	Matpower	**Case 14 Base**	**Case 14 Taps**
Cost (€/MWh)	8081.53	8081.53	8078.85

Table 2: Comparison of costs between different cases.

6.3 Effect of reactive control

Using the same base case, now the test is performed over the reactive power control. The limit is set as a maximum of a power factor of 0.8, which is translated to a maximum reactive power generation of 75% of the active power generation. Figure 13 shows the effect in the voltage values. There is a noticeable drop in magnitude in some of the buses.

In Figure 14, it can be seen that generator 2, which previously had a reactive generation of 24.12 Mvar, now has a generation of 20.96 Mvar, which corresponds exactly to the 75% of the active power generation. This generator being at its limit for reactive power generation means the others generator have to contribute to reactive generation, impacting the voltage levels in the grid.

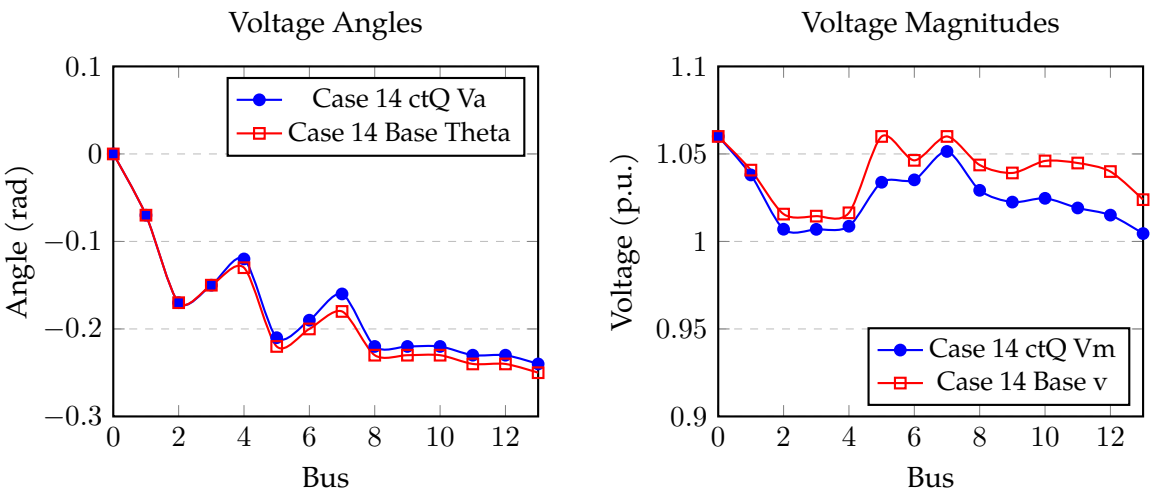

Figure 13: Voltage comparison for the case 14 with and without reactive control.

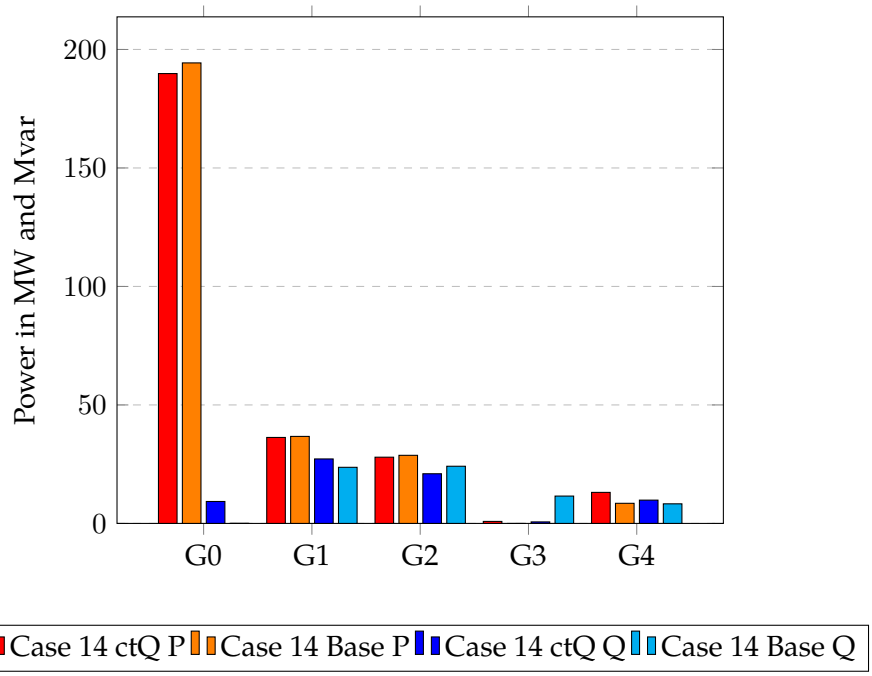

Figure 14: Generation comparison for the case 14 with and without reactive control.

The effect of this constraint can be seen in the cost value of the simulation. Table 3 shows an increase of around 0.075% in the cost of operation, which again can be significant in a bigger grid.

	Case 14 Base	**Case 14 ctQ**
Cost (€/MWh)	8081.53	8087.82

Table 3: Cost for Case 14 with Q control.

6.4 DC link and dual price study

The following study is performed over a small grid with two islands of three buses each. Both of them have the same configuration and elements, but their generators will have different costs profiles as follows:

$$\begin{aligned} &\text{Generator 1:} \quad c_{g1} = 1 + 2 \cdot P_{g1} \\ &\text{Generator 2:} \quad c_{g2} = 1 + 3 \cdot P_{g2} \\ &\text{Generator 3:} \quad c_{g3} = 1 + 1.5 \cdot P_{g3} \\ &\text{Generator 4:} \quad c_{g4} = 1 + 1 \cdot P_{g4} \end{aligned} \tag{48}$$

The grid is firstly tested without interconnection, obtaining the results for each island isolated. Then, a lossless DC link is added between the two islands. In Figure 15, the comparison between the generation profiles can be seen.

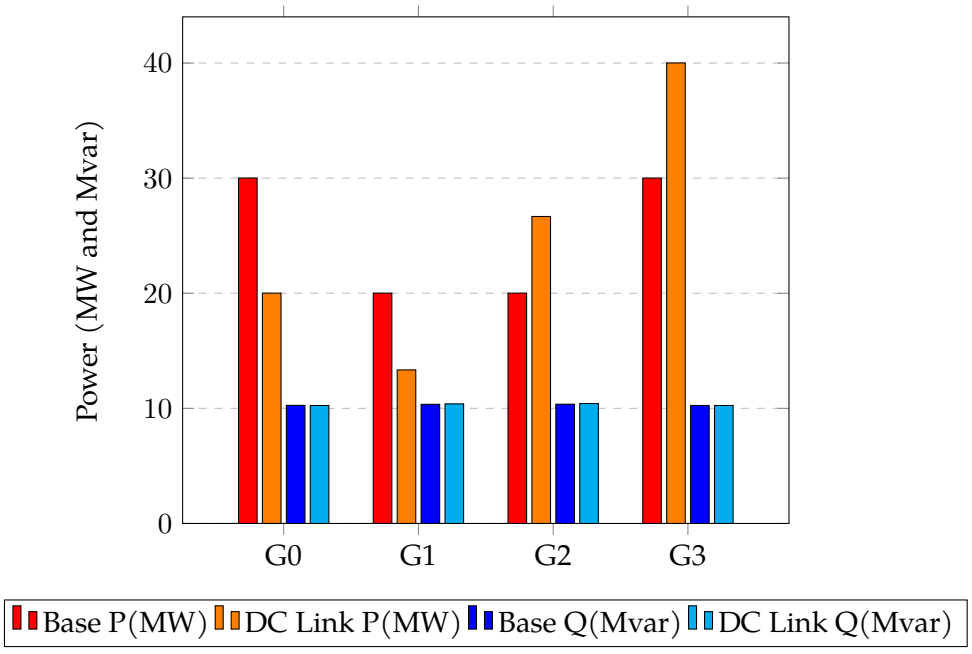

Figure 15: Generation comparison for the island case with and without DC link.

As expected, after both grids are connected, the generators of the second island become prevalent as their prices are lower than the first island, and the power is transferred through this DC link with a total power of $P_{DC} = 16.67$MW. The interconnection has a great impact in the dual price as seen in Figure 16.

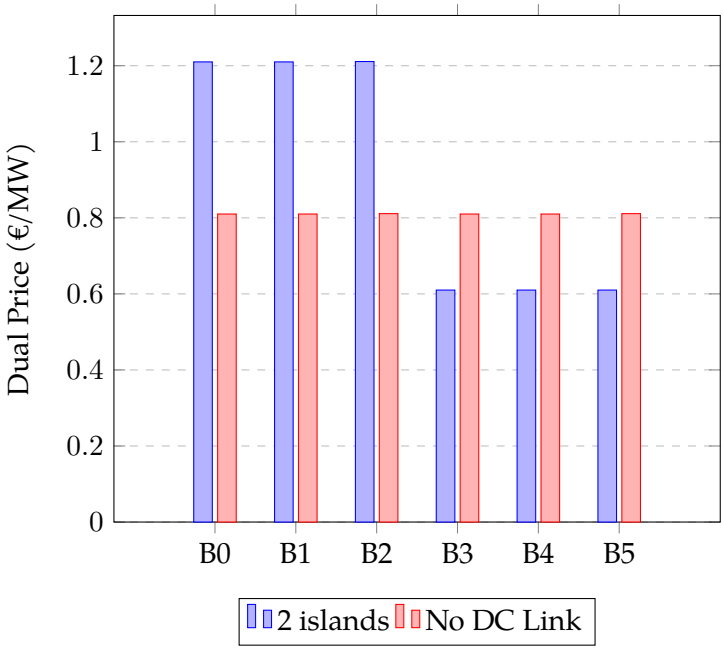

Figure 16: Dual price comparison for the case with and without DC link.

Finally, the cost of operation has obviously been reduced after connecting both islands as seen in table 4.

	Case 2 islands, No DC link	**Case 2 islands, DC link**
Cost (€/MWh)	4602.23	4102.12

Table 4: Costs for Case 2 islands.

6.5 Runtime comparison

Acknowledgments

A Environmental, Social, and Gender Impact

B Time Planning

C Budget

C.1 Equipment

The costs of machinery and digital tools required in the project development appear in Table 5.

Table 5: Equipment Costs.

Concept	Unit cost (€)	Quantity	Total (€)
Personal computer	1000.00	1	1000.00
Matlab individual annual license	2000.00	1	2000.00
Total			3000.00

C.2 Human resources

The working hours spent on the thesis and related work are captured in Table 6. It also includes the cost linked to the supervision process.

Table 6: Human Resources Costs.

Concept	Unit cost (€/h)	Quantity (h)	Total (€)
Research	25.00	100	2500.00
Code development	25.00	400	10000.00
Testing	25.00	150	3750.00
Writing	25.00	100	2500.00
Supervision	30.00	50	1500.00
Total			20250.00

C.3 Total budget

The total budget formed by aggregating equipment and human resources is shown in Table 7. No Value Added Tax (VAT) is considered in the budget.

2nd of July, 2023

Titouan Delorme

Table 7: Total Budget of the Thesis.

Concept	**Total (€)**
Equipment	3000.00
Human resources	20250.00
Total	23250.00