Santiago Peñate Vera

Practical Grid Modelling 2018

Contents

List of Figures

List of Tables

1 Introduction

This book aims at explaining grid modelling from a practical perspective, providing state of the art models, algorithms as well as implementation hints and examples.

The goal is to provide a reference document for researchers and computer engineers in the field of electric systems modelling, so that the task of building your own simulator or extend the already available open source ones becomes feasible.

The book assumes that the reader has a basic understanding of matrices, vectors and complex numbers. The notation used in the book is intended to be clear and accessible as much as it is practically possible.

Most of the formulas presented will imply vectorized operations such as matrix multiplications or element-wise multiplications. This notation makes it much easier to implement the numerical methods, and in case of scripting languages such as python, enables the use of fast linear algebra libraries. This is key in the implementation of electrical systems simulators, since most articles and books provide the formulas using summations, which implementation in languages other than C or $Fortran$ would make the simulator prohibitively slow in comparison with the vectorized version.

Vectorized formulas allow a simpler debugging and maintenance in the produced software.

So I really hope you enjoy this book and that find it useful for your purpose.

2 Notation

- $[A] \times [B] \rightarrow$ Matrix-Matrix dot product.
- $[A] \times [b] \rightarrow$ Matrix-Vector dot product.
- $[a] \times [b]^{\top} \rightarrow$ Vector-Vector dot product. The result is a number.
- $[A] \cdot [B] \rightarrow$ Matrix-Matrix element wise multiplication. A and B must have the same dimensions. The result is a matrix.
- $[a] \cdot [b] \rightarrow$ vector-vector element wise multiplication. The result is a Vector.
- $[A]^{*} \rightarrow$ Element-wise conjugate of the numbers composing $[A]$.
- $[A]^{-1} \rightarrow$ Matrix inverse.
- $[A]^{\top} \rightarrow$ Transposed Matrix or Vector.
- $[A]^{-1} \times [b] \rightarrow$ Solve the linear system where $[A]$ is the coefficients matrix and $[b]$ is the free terms vector. Never perform the inverse of $[A]$ and then multiply it by $[b]$. Instead use a linear system solver.
- $[A]_{(rows,:)} \rightarrow$ From the matrix $[A]$, pick the rows which indices are contained in the vector $rows$.
- $[A]_{(:,cols)} \rightarrow$ From the matrix $[A]$, pick the columns which indices are contained in the vector $cols$.

- $[A]_{(rows,cols)} \rightarrow$ From the matrix $[A]$, pick the rows which indices are contained in the vector $rows$ and the columns which indices are contained in the vector $cols$.
- $[b]_{(rows)} \rightarrow$ From the vector $[b]$, pick the elements contained in the vector or list $rows$.
- $diag([b]) \rightarrow$ Convert the vector $[b]$ into a diagonal matrix.
- $diag([A]) \rightarrow$ Extract the diagonal of the matrix $[A]$ as a vector.
- $Re\{[A]\} \rightarrow$ Extract the real part of $[A]$.
- $Im\{[A]\} \rightarrow$ Extract the imaginary part of $[A]$.
- $max([b]) \rightarrow$ Maximum value of the vector $[b]$.
- $max(c,d) \rightarrow$ Pick the greater value between c and d.
- $[1] \rightarrow$ Matrix of ones.
- $[0] \rightarrow$ Matrix of zeros.
- $[Idn] \rightarrow$ Identity matrix: Matrix of zeros with ones in the diagonal.

3 General network model

In this work, an unbalanced three-phase calculation engine is presented, being a reasonably general case. In the presented network model, the neutral wire and the earth "wire" are embedded into a three-phase equivalent using Kron's reduction. See [1] for a comprehensive explanation or simply [2] for practical application. Therefore the calculation will be performed for three phases (or two or one) at each calculation node.

The electrical grid can be assimilated to a graph. A graph is a mathematical object composed of nodes and edges (or branches). From a calculation point of view, a node is the place where the voltage is calculated given power and current injections and a branch is the place where the current and power that flows through it are computed given the nodes voltage.

A node (or Bus) has power injection or consumption devices attached to it. Examples are loads, generators, capacitor banks or any other device that injects or consumes power from the grid.

A branch might have devices attached that modify the flow through it. Such devices are known as FACTS (Flexible Alternating Current Transmission Systems).

A real life grid can be composed of several isolated groups of nodes (islands). It is impossible to perform calculations of several islands at once in the same numerical process. Hence, the maximal calculation object is the island circuit; A graph that does not contain further islands inside. In practice, a grid is split in it's islands. Each island is computed separately and the results are merged back to provide a consistent analysis interface of the whole grid.

[1] Florian Dorfler and Francesco Bullo. Kron reduction of graphs with applications to electrical networks. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 60 (1):150–163, 2013

[2] William H Kersting. *Distribution system modeling and analysis*. CRC press, 2012
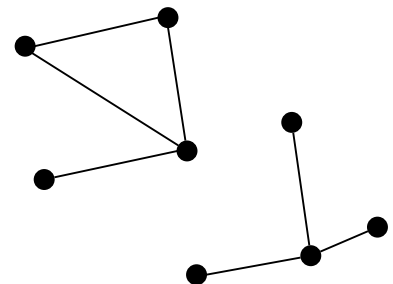
Figure 3.1: Graph with 8 nodes and 7 branches. The graph contains two islands.

4 Magnitudes and units

As the electric energy is mostly distributed in alternating current, electrical magnitudes are waves that vary their polarity (positive and negative value) and amplitude in time. because of this, the electrical magnitudes are expressed by complex numbers to denote the position of the value in the two-dimensional plane amplitude-time.

The units in electrical grid modelling are represented in the tables 4.1 and 4.2.

Figure 4.1: Voltage delay.

Magnitude	Unit	Recommended user input unit
Voltage	V (Volt)	kV (kilo-Volt)
Current	A (Ampere)	kA (kilo-Ampere)
Power	VA (Volt-Ampere)	MVA (Mega-Volt-Ampere)
Active power	W (Watt)	MW (Mega-Watt)
Reactive power	VAr (Volt-Ampere-reactive)	$MVAr$ (Mega-Volt-Ampere-reactive)
Impedance	Ω (Ohm)	Ω (Ohm) or per-unit
Admittance	S (Siemens)	S (Siemens) or per-unit

Table 4.1: Electrical magnitudes and their units.

The figure 4.1 shows two voltage waves. The one represented with a dotted line is delayed an angle θ with respect to the reference voltage wave represented by the plain black line. Since both waves are periodical, both can be represented as "phasors" or vectors indicating the magnitude's value and angle in the complex rectangular plane as depicted in the figure 4.2. Figures 4.1 and 4.2 are equivalent representations.

Figure 4.2: Voltage delay in phasor representation in the complex plane.

Magnitude	Real part	Imaginary part	Relation
S (Power)	P (Active power)	Q (Reactive power)	$S = P + jQ$
V (Voltage)	V_r (Real voltage)	V_i (Imaginary voltage)	$V = V_r + jV_i$
Expressed as			
	V_m (Voltage module)		
	θ (Voltage angle)		$V = V_m \cdot e^{\theta}$
I (Current)	I_r (Real current)	I_i (Imaginary current)	$I = I_r + jI_i$
Z (Impedance)	R (Resistance)	X (Inductance)	$Z = R + jX$
Y (Admittance)	G (Conductance)	B (Susceptance)	$Y = G + jB$

Table 4.2: Magnitudes and their real and imaginary complex components.

4.1 Components connection and their conversions

Let us assume a three-phase grid. The phases of the grid are denoted by the names of A, B and C. There are two main connection types that arise: Wye (like the letter "y") and Delta.

The wye and delta connections provide the ground to introduce the *phase* and *line* voltages. The phase to neutral voltage is called *phase volt-*

age, those are V_A, V_B and V_C. The phase to phase voltage is called *line voltage*, those are V_{AB}, V_{AC} and V_{BC}.

The delta connection   is depicted in the figure 4.3. The delta connection has no neutral.

Figure 4.3: Delta connection scheme.

The wye connection   is depicted in the figure 4.4. The star connection does have neutral (N). The wye connection is also known as star connection.
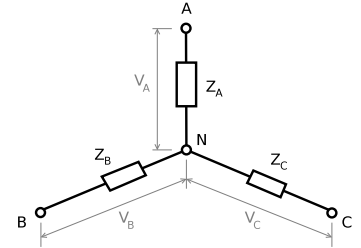
Figure 4.4: Wye connection scheme.

Delta to Wye ($\Delta \rightarrow Y$)   The transformation of a three phase connected shunt element in Delta to it's Wye equivalent is:

$$Elm_{Wye} = D \times Elm_{Delta} \tag{4.1}$$

Where D is:

$$D = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ -1 & 0 & 1 \end{bmatrix} \tag{4.2}$$

For instance, considering the impedances transformation from figures 4.3 and 4.4:

$$\begin{bmatrix} Z_A \\ Z_B \\ Z_C \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ -1 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} Z_{AB} \\ Z_{AC} \\ Z_{BC} \end{bmatrix} \tag{4.3}$$

Wye to Delta ($Y \rightarrow \Delta$)   The transformation of a three phase connected shunt element in Wye to it's Delta equivalent is:

$$Elm_{Delta} = D^{-1} \times Elm_{Wye} \tag{4.4}$$

Where D^{-1} is:

$$D^{-1} = \frac{1}{3} \begin{bmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \tag{4.5}$$

For instance, considering the impedances transformation from figures 4.3 and 4.4:

$$\begin{bmatrix} Z_{AB} \\ Z_{AC} \\ Z_{BC} \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \times \begin{bmatrix} Z_A \\ Z_B \\ Z_C \end{bmatrix} \tag{4.6}$$

4.2   Per unit system

In an electrical grid there are multiple levels of voltage. This situation introduces discontinuities in the numerical methods used to solve power flows and state estimations among others, producing an unstable convergence behaviour. To avoid this, the per unit system is introduced. A side effect of the per unit representation is to have a very convenient way of visualizing the grid magnitudes, all referenced to their base. In the per unit system, all the voltages are expressed in terms of their nominal value. In this case, all the grid voltage values are around one. For instance, a voltage value of 0.98 means that the voltage is 98 % of the nominal voltage value at that point.

For most exchange formats in computer programs, the element's magnitudes are expressed with a mix of actual units and per unit values. Regardless of this, a practical way of converting any electrical magnitude to its per unit equivalent is presented.

First, we must choose an arbitrary value of power base conversion. This value can be seen as the grid's nominal power, even though that concept is not related to any physical quantity, but it is rather a numerical artifice.

The base power is most commonly chosen to be $S_{Base} = 100 MVA$.

Magnitude	Base
V (Voltage)	V_{Base}: terminal's nominal voltage.
S (Power)	S_{Base}: Arbitrary value, usually 10 MVA.
I (Current)	$I_{Base} = S_{Base}/Vline_{Base} = S_{Base}/(V_{Base} \cdot \sqrt{3})$
Z (Impedance)	$Z_{Base} = V_{Base}^2/S_{Base}$
Y (Admittance)	$Y_{Base} = 1/Z_{Base}$

Table 4.3: Electrical magnitudes and their per unit base.

4.3   Sequence components simplification

Charles L. Fortescue presented in 1918 his famous article [1] in which he describes how to represent a three-phase element in the so-called *sequence components*.

The main use of this technique is to reduce the amount of impedances needed to represent a line or transformer from usually nine, to three (or even two) if the element is considered to be balanced. An element is considered balance if the impedance in all it's phases is equal and the phase-to-phase coupling impedances are also equal. This is an assumption that is commonly made for transmission grids (very high voltage) and distribution grids in high voltage. This advance allowed the popularization of the single-line diagrams in which every line represents a a number of wires transmitting power in a balanced scheme.

Fortesue defined a transformation matrix A_s and it's inverse as:

[1] Charles L Fortescue. Method of symmetrical co-ordinates applied to the solution of polyphase networks. *Transactions of the American Institute of Electrical Engineers*, 37(2):1027–1140, 1918

$$A_s = \begin{bmatrix} 1 & 1 & 1 \\ 1 & a^2 & a \\ 1 & a & a^2 \end{bmatrix} \tag{4.7}$$

$$A_s^{-1} = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 1 & a & a^2 \\ 1 & a^2 & a \end{bmatrix} \tag{4.8}$$

Where a is the transformation eigenvector:

$$a = 1^{120_{deg}} = 1 \cdot e^{j\frac{2}{3}\pi} = 1 \cdot cos\left(\frac{2}{3}\pi\right) + 1j \cdot sin\left(\frac{2}{3}\pi\right) \qquad (4.9)$$

$$a^2 = 1^{-120_{deg}} = 1 \cdot e^{-j\frac{2}{3}\pi} = 1 \cdot cos\left(\frac{2}{3}\pi\right) - 1j \cdot sin\left(\frac{2}{3}\pi\right) \qquad (4.10)$$

Then, any 3x3 impedance matrix representing the rectangular ABC three-phase impedance of an element (line, transformer, capacitor, etc.) can be transformed to a sequence equivalent using the formula:

$$Z_{seq} = A_s^{-1} \times Z_{ABC} \times A_s \qquad (4.11)$$

Example   Consider the following impedance matrix of a three-phase line. Example from [2].

[2] William H Kersting. *Distribution system modeling and analysis.* CRC press, 2012

$$Z_{ABC} = \begin{bmatrix} 0.4576 + j1.0780 & 0.1560 + j0.5017 & 0.1535 + j0.3849 \\ 0.1560 + j0.5017 & 0.4666 + j1.0482 & 0.1580 + j0.4236 \\ 0.1535 + j0.3849 & 0.1580 + j0.4236 & 0.4615 + j1.0651 \end{bmatrix}$$

Using equation 4.11, we obtain the sequence impedance matrix:

$$Z_{seq} = \begin{bmatrix} 0.7735 + j1.9373 & 0.0256 + j0.0115 & 0.0321 + j0.0159 \\ -0.0321 + j0.0159 & 0.3061 + j0.6270 & 0.0723j0.0060 \\ 0.0256 + j0.0115 & -0.0723j0.0059 & 0.3061 + j0.6270 \end{bmatrix}$$

For the sequence matrix, the non diagonal elements are neglected. Using only the three diagonal elements as:

$$Z_0 = 0.7735 + j1.9373$$
$$Z_1 = 0.3061 + j0.6270$$
$$Z_2 = 0.3061 + j0.6270$$

Observe that Z_1 and Z_2 are identical (with the shown numerical precision). It is very common in utilities to store only Z_0 and Z_1 to define a line. The balanced element assumption is very common and should be carefully used.

4.4   Building Z_{ABC} from the sequence components

Once the complete 3x3 impedance matrix has been reduced to the sequence components and only those have been stored in the utility database, the obtaining of the full 3x3 matrix might be necessary to perform unbalanced calculations. Of course we will not be able to obtain the exact original Z_{ABC} from the reduced sequence components, but the approximation is fair.

The approximated full impedance matrix is obtained from the sequence components as:

$$Z_{ABC_{approx}} = \frac{1}{3} \begin{bmatrix} 2Z_1 + Z_0 & Z_0 - Z_1 & Z_0 - Z_1 \\ Z_0 - Z_1 & 2Z_1 + Z_0 & Z_0 - Z_1 \\ Z_0 - Z_1 & Z_0 - Z_1 & 2Z_1 + Z_0 \end{bmatrix} \qquad (4.12)$$

If Z_2 would be significantly different from Z_1, the approximation is:

$$Z_{ABC_{approx}} = \begin{bmatrix} Z_0 + Z_1 + Z_2 & Z_0 + aZ_1 + a^2 Z_2 & Z_0 + a^2 Z_1 + aZ_2 \\ Z_0 + a^2 Z_1 + aZ_2 & Z_0 + Z_1 + Z_2 & Z_0 + aZ_1 + a^2 Z_2 \\ Z_0 + aZ_1 + a^2 Z_2 & Z_0 + a^2 Z_1 + aZ_2 & Z_0 + Z_1 + Z_2 \end{bmatrix}$$

(4.13)

Where the transformation eigenvector $a = e^{j2\pi/3}$ is used. This second approximation is used most in the transformation of the sequence impedances given for a synchronous machine to the ABC impedances used for calculation.

Example   Taking the calculated values from the previous section, we observe that Z_2 is close to Z_1 in the precision of interest, therefore we chose the first approximation. We need to compute two values before assembling the 3x3 matrix:

$$\frac{1}{3}(2 \cdot Z_1 + Z_0) = \frac{1}{3}(2 \cdot (0.3061 + j0.6270) + (0.7735 + j1.9373)) = 0.4619 + j1.0638$$

$$\frac{1}{3}(Z_0 - Z_1) = \frac{1}{3}((0.7735 + j1.9373) - (0.3061 + j0.6270)) = 0.1558 + j0.4368$$

The the approximated impedance matrix is:

$$Z_{ABC_{approx}} = \begin{bmatrix} 0.4619 + j1.0638 & 0.1558 + j0.4368 & 0.1558 + j0.4368 \\ 0.1558 + j0.4368 & 0.4619 + j1.0638 & 0.1558 + j0.4368 \\ 0.1558 + j0.4368 & 0.1558 + j0.4368 & 0.4619 + j1.0638 \end{bmatrix}$$

We observe that the calculation outcome is a fair approximation of the original Z_{ABC} and that the approximated matrix is symmetric, matching the balanced assumption, which the original impedance matrix did not fully comply with. Therefore, is the original Z_{ABC} was reduced assuming a balanced impedance distribution when that was not the case, if we build the approximated matrix, we will never know if it represents the reality.

5 Device models

In this chapter, the goal is to define the primitive matrices and vectors that represent the grid devices in the calculation circuit-wide matrices and vectors.
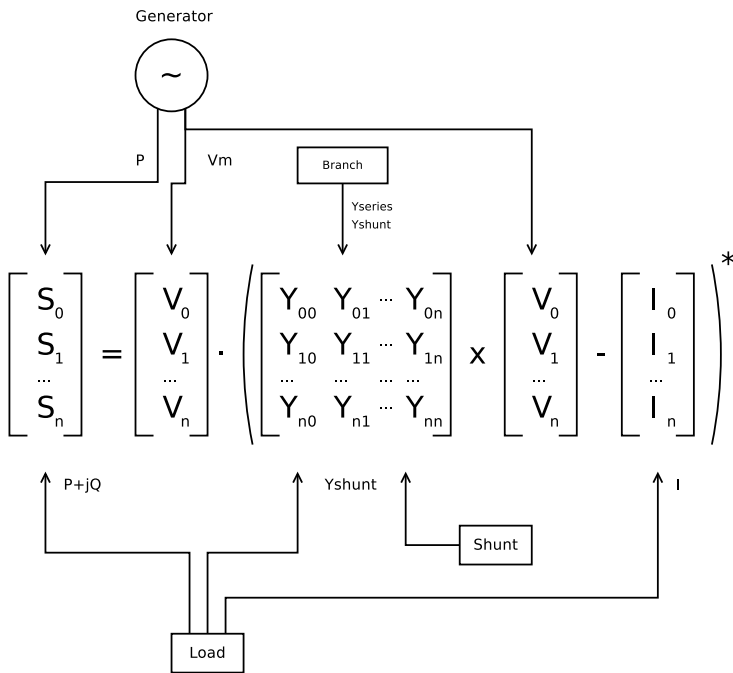
The branches will be represented as components of the circuit admittance matrix $[Y_{bus}]$. The loads, may be composed by constant power, constant current and constant admittance, those belong to $[S_{bus}]$, $[I_{bus}]$ and the diagonal of $[Y_{bus}]$ respectively. The voltage controlled devices such as generators, contribute only active power $[P]$ to $[S_{bus}]$ and voltage module $[Vm]$ to $[V_{bus}]$.

5.1 The bus and it's connected elements

The bus model is based upon the traditionally called *ZIP* bus model to indicate that the same model includes an impedance (Z), a current (I) and a power (P) value. To be consistent with our nomenclature, here we are going to refer to it as the $YISV$ model since at the node there is voltage(V), and a power (S), an admittance (Y) and a current (I) potentially connected.

The admittance, power and current values at the bus are derived from the elements connected to it (loads, generators, capacitor banks, etc.)

5.1.1   *Static device*

We can define two main device types. One with no voltage control (Static device) and one with voltage module control on its connection bus (Voltage controlled device). Devices such as capacitor banks, and some types of generators match the static device definition, hence those will inherit the definition provided here with the possibility of extending the definition with additional properties.

The modelling of a static device (i.e. load) is done as a composition of fixed impedance, fixed current and fixed power. This is known to match best the real behaviour of loads in electrical grids (Although in transmission they tend to be more fixed power-dominated). This is known as the ZIP model of a load. Here we are going to refer to it as ZIS, given that we actually specify the complex power (S) and not the real power (P).

Figure 5.2: ZIP / $YISV$ device model.

Param.	Description	Unit
name	Name of the device (string)	-
number of phases	Number of phases of the device (integer)	-
phases	Phases vector (i.e. [0, 1, 2] for ABC, [0,2] for AC, etc.)	-
conn	Connectivity mode (Y or Δ)	-
Z	Impedance (complex number)	MW, MVAr at the bus nominal voltage
I	Current (complex number)	MW, MVAr at the bus nominal voltage
S	Power (complex number)	MW, MVAr

The impedance Z is converted to admittance with $Y = 1/Z$.

When the systems matrix are forming, the admittance (Y) is added to the diagonal of the admittance matrix ($Ybus$ or simply Y), the power (S) os added to the power injections vector ($Sbus$ or simply S) and the current is added to the current injections vector ($Ibus$ or simply I). See the figure **??**.

A three phase admittance device such as a ZIP load or a capacitor bank has to be integrated into the nodal equation (see figure **??**)

Consider the model from the figure 5.3. The primitive admittance matrix is then:

In the case of a device connected in Δ, the primitive matrix is:

$$Y_{device} = \begin{bmatrix} y_{aa} & y_{ab} & y_{ac} \\ y_{ab} & y_{bb} & y_{bc} \\ y_{ac} & y_{bc} & y_{cc} \end{bmatrix} \tag{5.1}$$

Here, y_{ab}, y_{bc} and y_{ac} are given, and y_{aa}, y_{bb} and y_{cc} are computed as:

$$y_{aa} = y_{ab} + y_{ac}$$

Figure 5.3: Δ/Y shunt model.

$$y_{bb} = y_{ab} + y_{bc}$$

$$y_{cc} = y_{ac} + y_{bc}$$

In the case of a device connected in Y, the primitive matrix is:

$$Y_{device} = \begin{bmatrix} y_{aa} & 0 & 0 \\ 0 & y_{bb} & 0 \\ 0 & 0 & y_{cc} \end{bmatrix} \qquad (5.2)$$

The primitive matrices are added to the diagonal of the circuit admittance matrix $[Y_{bus}]$

5.1.2   *Voltage controlled device*

A voltage controlled generator is a generator that sets the voltage of the bus where it is connected.

The main parameters of a voltage controlled generator are:

Param.	Description	Unit
name	Name of the device (string)	-
number of phases	Number of phases of the device (integer)	-
phases	Phases vector (i.e. [0, 1, 2] for ABC, [0,2] for AC, etc.)	-
conn	Connectivity mode (Y or Δ)	-
V_{set}	Set voltage (number)	p.u.
P	Power (number)	MW
Q_{min}	Reactive power lower limit (number)	MVAr
Q_{max}	Reactive power upper limit (number)	MVAr

V=Vset
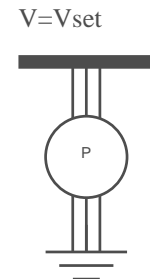
Figure 5.4: PV device model.

5.2   The branch element

To the effect of most calculations run in operation of a electrical system, the lines, transformers, and any other element that connects two nodes are represented by the so-called Π model.

5.2.1 Π Model

The pi model is composed by a series admittance Y_{serie} and a shunt admittance Y_{sh} divided in two. The shunt admittances are connected at the sending and receiving terminals (primary and secondary). To accommodate the possibility of regulating the voltage at the sending and/or receiving terminals, two per-unit transformers are included as well. The per unit transformers are modelled with the *tap ratio* parameters α and β.
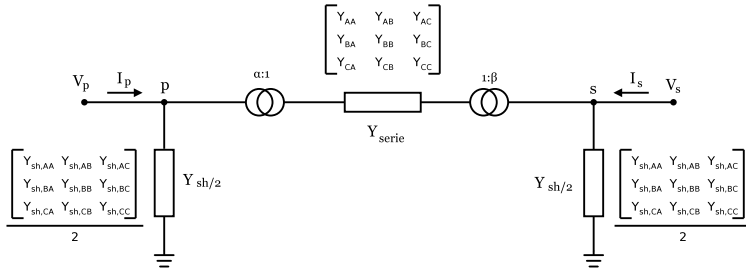
In the case of lines, the series admittance Y_{serie} is computed as the inverse of Z_{ABC}. From the line's calculation it is obtained the series impedance matrix Z_{ABC} and the shunt admittance matrix Y_{sh}.

Figure 5.5: General branch model.

The generalized admittance matrix that corresponds to the Π model is:

$$\begin{bmatrix} I_p \\ I_s \end{bmatrix} = \begin{bmatrix} Y_{pp} & Y_{ps} \\ Y_{sp} & Y_{ss} \end{bmatrix} \times \begin{bmatrix} V_p \\ V_s \end{bmatrix} \tag{5.3}$$

Where:

Y_{pp}	Y_{ps}	Y_{sp}	Y_{ss}
$\dfrac{Y_{series}+\frac{Y_{sh}}{2}}{\alpha^2}$	$\dfrac{-Y_{series}}{\alpha\beta}$	$\dfrac{-Y_{series}}{\alpha\beta}$	$\dfrac{Y_{series}+\frac{Y_{sh}}{2}}{\beta^2}$

Table 5.2: Equations of the generalized Π model.

The formula 5.3 will be used to compute the network admittance matrices.

5.2.2   Positive sequence model

Since the positive sequence simulation is very popular, a positive sequence model of the Π branch model is presented. The formulation of the admittance matrix of the branch is similar to the one presented for three phases, but each sub-matrix has only one element (one equivalent phase).

- z_{series}: Magnetizing impedance or simply series impedance. It is given in p.u.

- y_{shunt}: Leakage impedance or simply shunt impedance. It is given in p.u.

- tap_module: Module of the tap changer. It is a magnitude around 1.
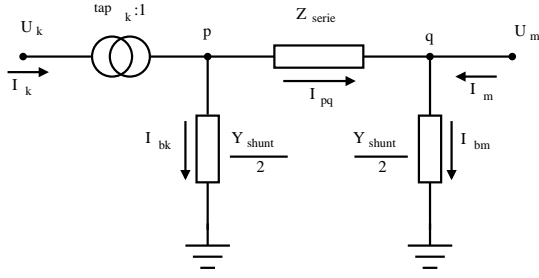
- tap_angle: Angle of the tap changer. Angle in radians.

In order to apply the effect of a branch to the admittance matrices, first
we compute the complex tap value.

Note that the tap value in the positive
sequence model is a complex value.
This is because the phase shifting that
occurs when using $\Delta - Y$ transformers is
accounted as a phase angle.

$$tap = tap\_module \cdot e^{-j \cdot tap\_angle}$$

Then we compose the equivalent series and shunt admittance values of
the branch. Both values are complex.

$$Y_s = \frac{1}{z_{series}}$$

$$Y_{sh} = \frac{y_{shunt}}{2}$$

- z_{series}: Series impedance of the branch composed by the line resistance
  and its inductance. $z_{series} = r + jl$

- y_{shunt}: Shunt admittance of the line composed by the conductance and
  the susceptance. $y_{shunt} = g + jb$

The general branch model is represented by a 2×2 matrix.

$$Y_{branch} = \left[\begin{array}{cc} Y_{ff} & Y_{ft} \\ Y_{tf} & Y_{tt} \end{array} \right]$$

In this matrix, the elements are the following:

$$Y_{ff} = \frac{Y_s + Y_{sh}}{tap \cdot conj(tap)}$$

$$Y_{ft} = -Y_s / conj(tap)$$

$$Y_{tf} = -Y_s / tap$$

$$Y_{tt} = Y_s + Y_{sh}$$

5.2.3   Line

The line element, utilizes the branch model as it has been formulated, only
that the use of the tap ratio relations α and β is not necessary.

Example   Let's assume that we have already computed the series impedance
and the shunt admittance of a three-phase line. The nominal voltage at
the line terminals is $66kV$ and we choose the base power to be $S_{base} =
100MVA$.

$$Z_{ABC} = \begin{bmatrix} 0.4576 + j1.0780 & 0.1560 + j0.5017 & 0.1535 + j0.3849 \\ 0.1560 + j0.5017 & 0.4666 + j1.0482 & 0.1580 + j0.4236 \\ 0.1535 + j0.3849 & 0.1580 + j0.4236 & 0.4615 + j1.0651 \end{bmatrix} \Omega$$

$$Y_{sh} = \begin{bmatrix} j5.6712 & -j1.8362 & -j0.7034 \\ -j1.8362 & j5.9774 & -j1.1690 \\ -j0.7034 & -j1.1690 & j5.3911 \end{bmatrix} \cdot 10^{-6} S$$

The first thing we need to do is to compute the base magnitudes:

$$Z_{base} = \frac{100MVA}{(66kV)^2} = 0.022956841 \quad \Omega$$

$$Y_{base} = \frac{1}{Z_{base}} = 43.56 \quad S$$

Then we must obtain the line series admittance matrix Y_{series} by inverting the 3x3 matrix Z_{ABC}. The we divide the resulting matrix by Y_{base}. Analogously we can invert the per unit impedance matrix:

$$Y_{series} = \left(\frac{Z_{ABC}}{Z_{base}} \right)^{-1} \quad p.u.$$

$$Y_{series} = \begin{bmatrix} 0.0112 - j0.0232 & -0.0054 + j0.008 & -0.0020 + j0.0052 \\ -0.0054 + j0.008 & 0.0125 - j0.024 & -0.0033 + j0.0063 \\ -0.0020 + j0.0052 & -0.0033 + j0.0063 & 0.0100 - j0.0224 \end{bmatrix} \quad p.u.$$

Dividing the shunt admittance by the base admittance we obtain the per unit shunt admittance:

$$Y_{sh} = \begin{bmatrix} j13.0193 & -j4.2153 & -j1.6148 \\ -j4.2153 & j13.7222 & -j2.6837 \\ -j1.6148 & -j2.6837 & j12.3763 \end{bmatrix} \cdot 10^{-8} \quad p.u.$$

Now we need to find the branch model admittances Y_{pp}, Y_{ps}, Y_{sp} and Y_{ss}.

$$Y_{pp} = Y_{ss} = Y_{series} + Y_{sh}/2 =$$

$$\begin{bmatrix} 0.0112 - j0.0232 & -0.0054 + j0.008 & -0.0020 + j0.0052 \\ -0.0054 + j0.008 & 0.0125 - j0.024 & -0.0033 + j0.0063 \\ -0.0020 + j0.0052 & -0.0033 + j0.0063 & 0.0100 - j0.0224 \end{bmatrix} \quad p.u.$$

$$(5.4)$$

$$Y_{ps} = Y_{sp} = -Y_{series} =$$

$$\begin{bmatrix} -0.0112 + j0.0232 & 0.0054 - j0.008 & 0.0020 - j0.0052 \\ 0.0054 - j0.008 & -0.0125 + j0.024 & 0.0033 - j0.0063 \\ 0.0020 - j0.0052 & 0.0033 - j0.0063 & -0.0100 + j0.0224 \end{bmatrix} \quad p.u.$$

$$(5.5)$$

If we convert the three-phase matrices into positive sequence values we obtain:

$$Y_{pp} = Y_{ss} = 0.0148 - j0.0296$$

$$Y_{ps} = Y_{sp} = -0.0148 + j0.0296$$

$$\begin{bmatrix} I_p \\ I_s \end{bmatrix} = \begin{bmatrix} 0.0148 - j0.0296 & -0.0148 + j0.0296 \\ -0.0148 + j0.0296 & 0.0148 - j0.0296 \end{bmatrix} \times \begin{bmatrix} V_p \\ V_s \end{bmatrix}$$

5.2.4 Transformer

The three phase transformer model implements the branch model as well, but the model admittances Y_{pp}, Y_{ps}, Y_{sp} and Y_{ss} vary depending on the connections types at the primary and at the secondary.

The most common transformer connections at the terminals are:

- Delta (Δ)

- Wye (Y)

- Grounded Wye (the neutral is grounded) (Yg)

Primary	Secondary	Y_{pp}	Y_{ss}	Y_{ps} and Y_{sp}
Yg	Yg	$\frac{1}{\alpha^2}Y_I$	$\frac{1}{\beta^2}Y_I$	$-\frac{1}{\alpha\beta}Y_I$
Yg	Y	$\frac{1}{3\alpha^2}Y_{II}$	$\frac{1}{3\beta^2}Y_{II}$	$-\frac{1}{3\alpha\beta}Y_{II}$
Yg	Δ	$\frac{1}{\alpha^2}Y_I$	$\frac{1}{\beta^2}Y_{II}$	$\frac{1}{\alpha\beta}Y_{III}$
Y	Yg	$\frac{1}{3\alpha^2}Y_{II}$	$\frac{1}{3\beta^2}Y_{II}$	$-\frac{1}{3\alpha\beta}Y_{II}$
Y	Y	$\frac{1}{3\alpha^2}Y_{II}$	$\frac{1}{\beta^2}Y_{II}$	$\frac{1}{\alpha\beta}Y_{III}$
Δ	Δ	$\frac{1}{\alpha^2}Y_{II}$	$\frac{1}{\beta^2}Y_{II}$	$-\frac{1}{\alpha\beta}Y_{II}$
Δ	Yg	$\frac{1}{\alpha^2}Y_{II}$	$\frac{1}{\beta^2}Y_I$	$\frac{1}{\alpha\beta}Y_{III}$

Table 5.3: Three-phase transformer impedances of the branch model.

The table 5.3 lays out the branch admittances for every pair of connections. The source for this model is the excellent book by J.Arrillaga[1].

[1] Jos Arrillaga and CP Arnold. *Computer analysis of power systems*. Wiley Online Library, 1990

$$Y_I = \begin{bmatrix} y_t & 0 & 0 \\ 0 & y_t & 0 \\ 0 & 0 & y_t \end{bmatrix} \tag{5.6}$$

$$Y_{II} = \begin{bmatrix} 2y_t & -y_t & -y_t \\ -y_t & 2y_t & -y_t \\ -y_t & -y_t & 2y_t \end{bmatrix} \tag{5.7}$$

$$Y_{III} = \begin{bmatrix} -y_t & y_t & 0 \\ 0 & -y_t & y_t \\ y_t & 0 & -y_t \end{bmatrix} \tag{5.8}$$

y_t is the transformer winding per unit admittance. It is given in values per units from the transformer specifications sheet. Usually it is given either directly as magnetizing resistance r_m and inductance x_m, in which case:

$$y_t = \frac{3}{r_l + jx_l} \tag{5.9}$$

Or it is given as the "short circuit study" values. This is a more complete case.

5.2.5   Transformer definition from the short circuit study

In order to get the series impedance and shunt admittance of the transformer to match the branch model, it is advised to transform the specification sheet values of the device into the desired values. The values to take from the specifications sheet are:

- S_n: Nominal power in MVA.

- U_{hv}: Voltage at the high-voltage side in kV.

- U_{lv}: Voltage at the low-voltage side in kV.

- U_{sc}: Short circuit voltage in %.

- P_{cu}: Copper losses in kW.

- I_0: No load current in %.

- GX_{hv1}: Reactance contribution to the HV side. Value from 0 to 1.

- GR_{hv1}: Resistance contribution to the HV side Value from 0 to 1.

Then, the series and shunt impedances are computed as follows:

Nominal impedance HV (Ohm):

$$Zn_{hv} = U_{hv}^2/S_n \tag{5.10}$$

Nominal impedance LV (Ohm):

$$Zn_{lv} = U_{lv}^2/S_n \tag{5.11}$$

Short circuit impedance (p.u.):

$$z_{sc} = U_{sc}/100 \tag{5.12}$$

Short circuit resistance (p.u.):

$$r_{sc} = \frac{P_{cu}/1000}{S_n} \tag{5.13}$$

Short circuit reactance (p.u.):

$$x_{sc} = \sqrt{z_{sc}^2 - r_{sc}^2} \tag{5.14}$$

HV resistance (p.u.):

$$r_{cu,hv} = r_{sc} \cdot GR_{hv1} \tag{5.15}$$

LV resistance (p.u.):

$$r_{cu,lv} = r_{sc} \cdot (1 - GR_{hv1}) \tag{5.16}$$

HV shunt reactance (p.u.):

$$xs_{hv} = x_{sc} \cdot GX_{hv1} \tag{5.17}$$

LV shunt reactance (p.u.):

$$xs_{lv} = x_{sc} \cdot (1 - GX_{hv1}) \tag{5.18}$$

Shunt resistance (p.u.):

$$r_{fe} = \frac{Sn}{P_{fe}/1000} \tag{5.19}$$

Magnetization impedance (p.u.):

$$z_m = \frac{1}{I_0/100} \tag{5.20}$$

Magnetization reactance (p.u.):

$$x_m = \frac{1}{\sqrt{\frac{1}{z_m^2} - \frac{1}{r_{fe}^2}}} \qquad (5.21)$$

If the content of the square root is negative, set the magnetization impedance to zero.

The final complex calculated parameters in per unit are:

Magnetizing impedance (or series impedance):

$$z_{series} = Z_m = r_{sc} + j \cdot x_{sc} \quad (5.22)$$

The series admittance is [p.u.]:

$$y_{series} = \frac{1}{z_{series}} \qquad (5.23)$$

Leakage impedance (or shunt impedance):

$$Z_l = r_{fe} + j \cdot x_m \qquad (5.24)$$

Shunt admittance [p.u.]:

$$y_{shunt} = 1/Z_l \qquad (5.25)$$

The series admittance for the three-phase model [p.u.]:

$$y_t = \frac{3}{z_{series}} \qquad (5.26)$$

we divide the impedance by 3, to reflect the three phases.

Example Let's consider a distribution transformer with the following name-plate characteristics:

- Primary connection: Δ
- Secondary connection: Yg
- $S_n = 0.5 MVA$
- $U_{hv} = 20 kV$
- $U_{lv} = 0.4 kV$
- $U_{sc} = 6\%$

- $p_{cu} = 6 kW$
- $p_{fe} = 1.4 kW$
- $I_0 = 0.28\%$
- $GR_{hv} = 0.5$
- $GX_{hv} = 0.5$

First we obtain the impedance value, from the short circuit study:

$$Zn_{hv} = 800 \quad \Omega$$
$$Zn_{lv} = 0.3200 \quad \Omega$$
$$z_{sc} = 0.0600 \quad p.u.$$
$$r_{sc} = 0.0120 \quad p.u.$$
$$x_{sc} = 17.0103 \quad p.u.$$
$$r_{cu,hv} = 0.0060 \quad p.u.$$
$$r_{cu,lv} = 0.0060 \quad p.u.$$
$$x_{s,hv} = 8.5052 \quad p.u.$$

$$x_{s,lv} = 8.5052 \quad p.u.$$
$$r_{fe} = 357.1429 \quad p.u.$$
$$z_m = 357.1429 \quad p.u.$$
$$x_m = 24296003999.8084 \quad p.u.$$
$$z_{series} = 0.0120 + j17.0103 \quad p.u.$$
$$y_{series} = 0.0000 - j0.0588 \quad p.u.$$
$$y_{shunt} = 0.0000 - j0.0000 \quad p.u.$$
$$y_t = 0.0001 - j0.1764 \quad p.u.$$

Once the we have obtained the y_t value, we start building the appropriate branch impedance matrices Y_{pp}, Y_{ps}, Y_{sp} and Y_{ss}.

$$Y_I = \begin{bmatrix} 0.0001 - j0.1764 & 0.0000 + j0.0000 & 0.0000 + j0.0000 \\ 0.0000 + j0.0000 & 0.0001 - j0.1764 & 0.0000 + j0.0000 \\ 0.0000 + j0.0000 & 0.0000 + j0.0000 & 0.0001 - j0.1764 \end{bmatrix}$$

$$Y_{II} = \begin{bmatrix} 0.0002 - j0.3527 & -0.0001 + j0.1764 & -0.0001 + j0.1764 \\ -0.0001 + j0.1764 & 0.0002 - j0.3527 & -0.0001 + j0.1764 \\ -0.0001 + j0.1764 & -0.0001 + j0.1764 & 0.0002 - j0.3527 \end{bmatrix}$$

$$Y_{III} = \begin{bmatrix} -0.0001 + j0.1764 & 0.0001 - j0.1764 & 0.0000 + j0.0000 \\ 0.0000 + j0.0000 & -0.0001 + j0.1764 & 0.0001 - j0.1764 \\ 0.0001 - j0.1764 & 0.0000 + j0.0000 & -0.0001 + j0.1764 \end{bmatrix}$$

According to the transformer $\Delta \rightarrow Yg$ connection, we build the branch model admittances using the formulas for the table 5.3:

$$Y_{pp} = \begin{bmatrix} 0.0002 - j0.3527 & -0.0001 + j0.1764 & -0.0001 + j0.1764 \\ -0.0001 + j0.1764 & 0.0002 - j0.3527 & -0.0001 + j0.1764 \\ -0.0001 + j0.1764 & -0.0001 + j0.1764 & 0.0002 - j0.3527 \end{bmatrix}$$

$$Y_{ss} = \begin{bmatrix} 0.0001 - j0.1764 & 0.0000 + j0.0000 & 0.0000 + j0.0000 \\ 0.0000 + j0.0000 & 0.0001 - j0.1764 & 0.0000 + j0.0000 \\ 0.0000 + j0.0000 & 0.0000 + j0.0000 & 0.0001 - j0.1764 \end{bmatrix}$$

$$Y_{ps} = Y_{sp} =$$

$$\begin{bmatrix} -0.0001 + j0.1764 & 0.0001 - j0.1764 & 0.0000 + j0.0000 \\ 0.0000 + j0.0000 & -0.0001 + j0.1764 & 0.0001 - j0.1764 \\ 0.0001 - j0.1764 & 0.0000 + j0.0000 & -0.0001 + j0.1764 \end{bmatrix}$$

There is no computational need to assemble the 6x6 element impedance matrix in a computer program. The branch model matrices are added to the full circuit matrix instead.

5.2.6   *Voltage regulator*

Since all the electrical models are in per-unit values, the voltage regulator is modelled in the exact same way as a transformer.

6 Topology analysis and consolidation

In Energy Management Systems Software (EMS) the topology is specified using devices as connectivity nodes, terminals, jumpers, branches, etc. These topological objects must be converted into a Bus-Branch model in order to be able to compute power flows and other electrical calculations.

6.1 Topology processing

The topological process presented here is efficient and can render both the grid connectivity and the calculation matrices (Y_{Bus}, S_{bus}, etc.) using linear algebra as much as possible.
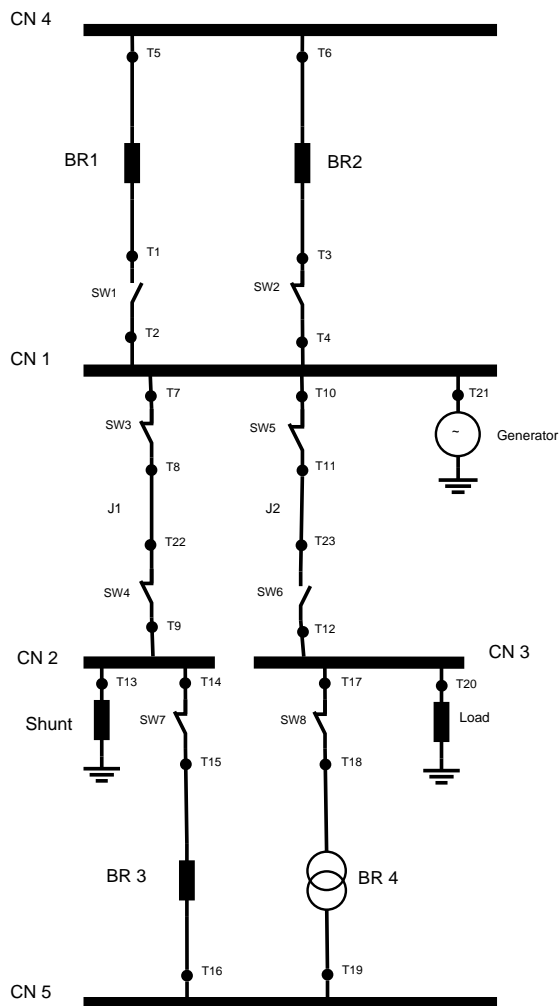

Figure 6.1: Substation example with all the connectivity representation in the CIM / EMS sense. On the right side there is the reduction of the topology to a calculation model.

The elements presented in the example of the figure 6.1 are:

- Connectivity Node (CN): This is a physical joint of devices. It is usually assimilated to a bus bar.

- Terminal (T): This element represents the terminal of a device. It can be regarded as a joining of two devices.

- Switch (SW): A device to interrupt the flow of electric current. There may be many more variants of this device.

- Branch (BR): Any element used to transport power.

- Shunt (SH): Any device used to inject or subtract power (loads, generators, etc.)

For calculation, the connectivity is solved first, originating one or more calculation buses. The switches, terminals, etc. are irrelevant at calculation level, but they are very relevant in the definition of the grid topology prior to the calculation. We shall start by forming the connectivity matrices of each type of device with the terminal devices. From those connectivity primitives, we are able to form any connectivity matrix in the circuit.

The following primitive matrices are formed by element inspection. This is, going through the objects stored in memory and inspecting the terminals that correspond to each one of them.

The simplest way to do this is to go through the terminals first and construct a dictionary to relate the terminal ID with the terminal position index in the terminals list of objects. This dictionary is later used to get the terminal index of each of the other elements of the grid.

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20	T21	T22	T23
CN1		1		1			1			1												1	
CN2									1				1	1									
CN3												1					1		1				
CN4				1	1																		
CN5																1		1					

Table 6.1: Connectivity matrix of the connectivity nodes (Buses) with the terminals: $[C_{bus,terminal}]$

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20	T21	T22	T23
BR1					1																		
BR2						1																	
BR3															1								
BR4																	1						
J1								1															
J2											1												

Table 6.2: Connectivity matrix of the branch objects with the terminals at the "from" side: $[C_{branch,terminal\_f}]$

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20	T21	T22	T23
BR1	1																						
BR2			1																				
BR3															1								
BR4																	1						
J1																					1		
J2																							1

Table 6.3: Connectivity matrix of the branch objects with the terminals at the "to" side: $[C_{branch,terminal\_t}]$

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20	T21	T22	T23
SW1	1	1																					
SW2			1	1																			
SW3							1	1															
SW4									1												1		
SW5						1	1																
SW6												1											1
SW7														1	1								
SW8																			1	1			

Table 6.4: Connectivity matrix of the switch objects with the terminals: $[C_{switch,terminal}]$

	SW1	SW2	SW3	SW4	SW5	SW6	SW7	SW8
States	0	1	1	1	1	0	1	1

Table 6.5: Switch states (0: open, 1: closed): $[switch\_states]$

We need as many primitive matrices as device types there are in the grid. Once the primitive matrices of all the devices are formed we can start the topological processing by multiplying these primitive matrices. We shall start by modifying the switches connectivity matrix with the switch states vector:

$$[C_{switch\_state,terminal}] = [C_{switch,terminal}] \times diag([switch\_states]) \quad (6.1)$$

This operation causes the switches open/close state to be applied to the topological calculations. Next, compute the connectivity matrix of the branches and the switches:

$$[C_{branch,switch\_f}] = [C_{branch,terminal\_f}] \times [C_{switch\_state,terminal}]^{\top} \quad (6.2)$$

$$[C_{branch,switch\_t}] = [C_{branch,terminal\_t}] \times [C_{switch\_state,terminal}]^{\top} \quad (6.3)$$

Then, compute the connectivity matrix of the connectivity nodes (buses) and the switches:

$$[C_{bus,switch}] = [C_{bus,terminal}] \times [C_{switch\_state,terminal}]^{\top} \quad (6.4)$$

Compute the connectivity matrix of the Branches and the connectivity nodes (buses):

$$[C_f] = [C_{branch,switch\_f}] \times [C_{bus,switch}]^{\top} \quad (6.5)$$

$$[C_t] = [C_{branch,switch\_t}] \times [C_{bus,switch}]^{\top} \quad (6.6)$$

These connectivity matrices are used for calculation since in the numerical methods we only care about buses and branches.

	CN1	CN2	CN3	CN4	CN5
BR1				1	
BR2				1	
BR3		1			
BR4			1		
J1	1				
J2	1				

Table 6.6: Calculated connectivity matrix of branches at the "from" side with the Connectivity nodes (Buses): $[C_f]$

The graph of the circuit is automatically constructed by computing the Bus-Bus adjacency matrix $[C_{bus,bus}]$:

$$[C_{bus,bus}] = [C_{bus,switch}] \times [C_{bus,switch}]^{\top} \quad (6.7)$$

At this point the topological process has to continue by detecting if any island has been formed.

	CN1	CN2	CN3	CN4	CN5
BR1					
BR2	1				
BR3					1
BR4					1
J1		1			
J2					

Table 6.7: Calculated connectivity matrix of branches at the "to" side with the Connectivity nodes (Buses): $[C_t]$

	CN1	CN2	CN3	CN4	CN5
CN1	1	1		1	
CN2	1	1			1
CN3			1		1
CN4	1			1	
CN5		1	1		1

Table 6.8: Grid adjacency matrix $[C_{bus,bus}]$, also known as A. Serves as a graph to detect the islands formed.

6.2 Islands detection

The admittance matrix of a circuit with more than one island is singular. Therefore, the circuit has to be split in sub-circuits in order to be solved. The suggested algorithm to find the islands of a circuit is the Depth First Search (DFS) algorithm.
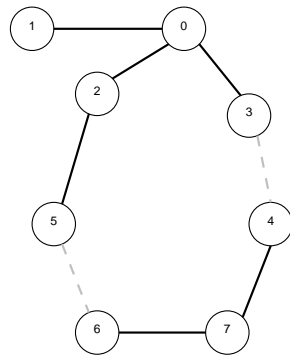
Figure 6.2: Graph with two islands.

Previously it was already determined that the circuit complete graph is given by the Bus-Bus connectivity matrix $[C_{bus,bus}]$. This matrix is also known as the node adjacency matrix. For algorithmic purposes we will call it the adjacency matrix A. As a side note, the matrix A is a sparse matrix. See [1].

The algorithm to find the circuit islands is:

1. N is the number of nodes (buses) of the circuit.

2. Declare `visited` as a vector of length N.

3. Declare `islands` as a list of lists.

4. Declare `island_idx = 0`.

5. For $i = 0$ to N:

   (a) If `visited[i] = False`

      i. If `island_idx` is greater or equal than the amount of islands in `islands`, initialize a new island as an empty list into `islands`.
      // Next we need to run a DFS process for the current node i.

[1] Compressed column storage (ccs). URL `http://netlib.org/linalg/html_templates/node92.html`

For algorithmic purposes, A it is chosen to be a CSC sparse matrix. This is important because the following algorithm uses the CSC sparse structure to find the adjacent elements of a node.

    ii. Declare `stack` as a Stack cue.

    iii. Append the node i to the stack.

    iv. While the `stack` is not empty:

        A. Pick the first element of the stack: `v = stack.pop(0)`

        B. Add the picked element to the current island: `islands[island_idx].append(v)`

        C. For `j=A.col_ptr[v]` to `A.col_ptr[v+1]`:

           • Declare `k=A.row_ind[j]`. Here k is the row index in the CSC sparse storage scheme.

           • If `visited[k] = false` then add k to the stack.

        D. End of the for loop.

    v. End of the while loop.

    vi. Increase the island index: `island_idx += 1`

6. End of the for loop.

7. End of the algorithm

At the end of the algorithm we have a list where each element is a list of bus indices composing a circuit island. With this information we are able to split the singular circuit matrices into non-singular matrices in order to be able to compute the electrical calculations. Needless to say that if only only island is found, the circuit is already one island and no matrix subdivision is needed.

6.3 Finding the branch indices that correspond to an island

When splitting the circuit into island it is not enough to know the nodes that compose that island, we need to know which branches fall into each island as well. To accomplish that, we can define the following branch-bus connectivity matrix:

$$[C] = [C_f] + [C_t] \tag{6.8}$$

Say we open the switch SW7 of figure 6.1. The resulting Branch-Bus connectivity matrix is:

	CN1	CN2	CN3	CN4	CN5
BR1				1	
BR2	1			1	
BR3					1
BR4			1		1
J1	1	1			
J2	1				

Table 6.9: Calculated connectivity matrix of branches with the Connectivity nodes (Buses): $[C]$

If we select the columns that match with the particular indices of an island, we get:

$$[C_{island}] = [C]_{(:,island)} \tag{6.9}$$

One of the two islands formed when opening SW7 is the one composed by the buses CN1, CN2 and CN4. The matrix $[C_{island}]$ for that island is:

	CN1	CN2	CN4	column summation
BR1			1	1
BR2	1		1	2
BR3				0
BR4				0
J1	1	1		2
J2	1			1

Table 6.10: Calculated connectivity matrix of branches with the Connectivity nodes (Buses): $[C]$

Performing the summation of the rows of $[C_{island}]$ we know that if a row summation is greater than zero, then the branch belongs to the island. In this case we may conclude that BR1, BR2, J1 and J2 belong to the first island.

6.4   Putting it all together: Composing calculation matrices and vectors

Chapter 5 discussed how to create matrix primitives to represent some devices present in the grid. At the beginning of this chapter, it was discussed how to process the topological information into connectivity matrices. Here we are going to explain how to merge both in order to compose the final calculation matrices and vectors.

For the computations ahead we will need to have a number of admittance-based matrices. We will compose matrices and vector for the complete circuit, and then split those per circuit island. Remember that the calculation matrices are only valid for island circuits, this is because multi-island circuits present singular admittance matrices that lead to no numerical solution by their own.
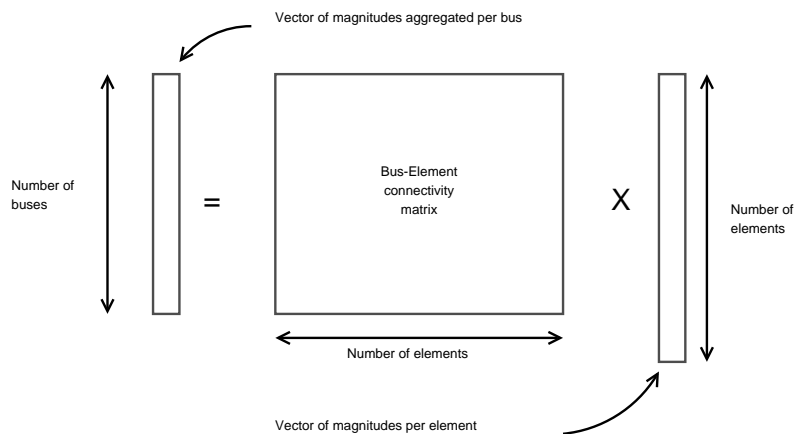
Figure 6.3: Element bus aggregations.

The logic is to have a vector of magnitudes of an object type (i.e. all the active power power for all the generators) and a connectivity matrix that relates the generators with the buses of the grid. Multiplying the connectivity matrix by the element magnitudes vector we obtain the buses magnitudes vector that we need for calculation.

Power, current and voltage vectors

$$[demand\_power] = [C_{load,bus}]^{\top} \cdot [load\_power\_S] \qquad (6.10)$$

$$[generation\_power] = [C_{generation,bus}]^{\top} \cdot [generation\_power] \qquad (6.11)$$

$$[S_{bus}] = [generation\_power] - [demand\_power] \qquad (6.12)$$

$$[I_{bus}] = -[C_{load,bus}]^{\top} \cdot [load\_power\_I] \qquad (6.13)$$

Admittance matrix

$$[Ysh] = [C_{shunt,bus}]^{\top} \cdot [shunt\_power] + [C_{load,bus}]^{\top} \cdot [load\_power\_Y]$$

$$(6.14)$$

$$Y_f = diag([BR_{yff}]) \times [C_f] + diag([BR_{yft}]) \times [C_t] \qquad (6.15)$$

$$Y_t = diag([BR_{ytf}]) \times [C_f] + diag([BR_{ytt}]) \times [C_t] \qquad (6.16)$$

$$Ybus = [C_f]^{\top} \times [Y_f] + [C_t]^{\top} \times Y_t + diag([Ysh]) \qquad (6.17)$$

$[BR_{yff}]$, $[BR_{yft}]$, $[BR_{ytf}]$ and $[BR_{ytt}]$ are vectors of length equal to the number of branches, where the branch primitive admittance matrix components are stored. This way we can leverage the connectivity matrices $[C_f]$ and $[C_t]$ to form $[Y_{bus}]$

7 Power flow

The power flow problem consists in finding the node voltages that correspond to the injected current and power values. The fundamental equation to be solved is:

$$[S] = [V] \times ([Y] \times [V] - [I])^*$$ (7.1)

Where:

- $[S]$: Vector of power injections.

- $[I]$: Vector of current injections.

- $[V]$: Vector of voltages.

- $[Y]$: Admittance matrix.

Note that the equation 7.1 is non-linear when solving for the voltage (V). However if there are no power injections and only current injections, the equation becomes linear. Because of the non-linearity, we will be using iterative methods to solve this equation.

7.1 Types of buses

At each node, we can consider the following magnitudes to exist:

- Vm: Voltage module.

- θ: Voltage angle.

- P: Active power injection/consumption.

- Q: Reactive power injection/consumption.

In some nodes we know the active power injection (generation nodes), in some nodes we just know the consumption power (load nodes), and in some nodes we artificially define the voltage (slack nodes). The power flow problem is formulated such that in every node two variables are set and the other two are computed. Thus we can define the node types as:

Bus type	Vm	θ	P	Q
PQ	Calc.	Calc.	Set	Set
PV	Set	Calc.	Set	Calc.
VD (slack)	Set	Set	Calc.	Calc.

Table 7.1: Bus types.

There must be at least one VD or slack node in order to be able to compute the power flow problem.

7.2 Z-Matrix

The Z-Matrix method is a simple iterative method that provides good results, especially in distribution grids where the voltage variations are not large.

The Z-Matrix method requires the Kron reduction of the slack nodes of the circuit. This reduction is explained below:
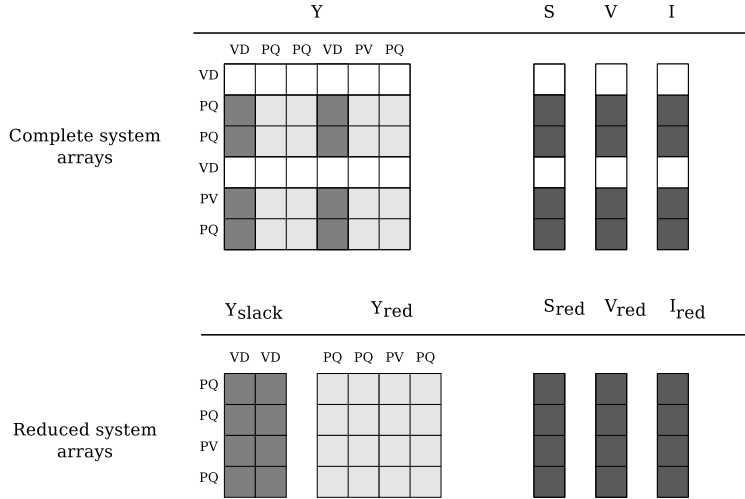
Figure 7.1: Circuit's admittance matrix and vectors reduction. Representation for a six node circuit with two slack nodes.

Of course, not only the admittance matrix has to be reduced. The associated vectors need to be reduced as well by removing the slack related values. The slack related values of the voltage and the admittance matrix are kept for operation.

$$[Y_{red}] = [Y]_{(pqpv,pqpv)} \tag{7.2}$$

$$[Y_{slack}] = [Y]_{(pqpv,vd)} \tag{7.3}$$

$$[V_{red}] = [V]_{(pqpv)} \tag{7.4}$$

$$[V_{slack}] = [V]_{vd} \tag{7.5}$$

$$[S_{red}] = [S]_{(pqpv)} \tag{7.6}$$

$$[I_{red}] = [I]_{(pqpv)} \tag{7.7}$$

Once we have all the reduced magnitudes, it means that we have *removed* the slack (VD) nodes from the circuit. However we must keep the slack nodes influence in the rest of the nodes in the form of current injections ($[I_{slack}]$).

First we copy $[V_{red}]$ into another variable $[V_{prev}]$.

$$[V_{prev}] = [V_{red}] \tag{7.8}$$

The current injections appearing by the removal of the slack nodes are:

$$[I_{slack}] = [Y_{slack}] \times [V_{slack}] \tag{7.9}$$

The voltages that arise from the slack current injections are:

$$[C_k] = [Y_{red}] \times [I_{slack}] \tag{7.10}$$

The total injected currents including the power injections are:

$$[I_k] = \frac{[S_{red}]}{[V_{prev}]} + [I_{red}] \tag{7.11}$$

The next step is to compute the nodes voltage due to all the power and current injections:

$$[V_k] = [Y_{red}]^{-1} \times [I_k] - [C_k] \tag{7.12}$$

Do not invert $[Y_{red}]$. Instead factorize it once, and use the factorization.

Now the error is the infinite norm of the voltage difference.

$$error = ||[V_{prev}] - [V_k]||_\infty = max(abs([V_{prev}] - [V_k])) \tag{7.13}$$

The infinite norm of the voltage difference between the previous iteration and the current iteration is a poor convergence criteria, but it is usually the only working criteria in practice for this method.

Now we need to correct the voltage for the PV nodes so that they control the voltage module.

$$[V_k]_{(pv_{red})} = [V_k] \cdot \frac{|[V^{esp}]_{(pv_{red})}|}{|[V_k]_{(pv_{red})}|} \tag{7.14}$$

Note that pv_{red} is the vector of pv node indices in the reduced scheme. It is not equal to the pv indices vector.

Next, we copy the voltage solution to the previous voltage solution vector:

$$[V_{prev}] = [V_k] \tag{7.15}$$

Repeat equations 7.11 to 7.15 until the error is less or equal to a given tolerance not too strict like 1×10^{-3}.

Finally, since we have found the voltages for the PQ and PV nodes only, we should return a voltage vector with the voltages for all the nodes. Because of this, we copy the voltage solution for the reduced system into a copy of the original voltage vector.

$$[V]_{(pqpv)} = [V_k] \tag{7.16}$$

7.3   Jacobian based power flow

The derivative based methods are more accurate than the derivative free methods. They are usually more robust at the cost of being more computationally expensive. In this section we'll start by introducing the Jacobian matrix. It is the derivative of the power flow equation [7.1] with respect to a set of voltage values. The section continues by introducing the methods that use this matrix to solve the power flow problem.
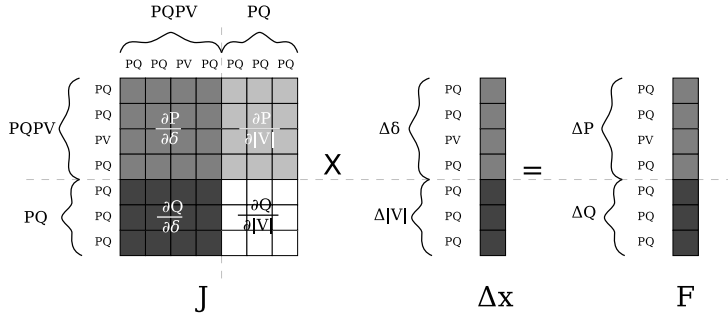

Figure 7.2: Jacobian based linear system to compute the voltage increments.

The Jacobian linear system ($[J] \times [\Delta x] = [F]$), allows the computation of the voltage increment $[\Delta x]$ to apply to the voltage solution $[V]$ in order to achieve a voltage that solves the power flow equation 7.1. This process is iterative and recursive, hence to have a initial voltage solution in the vicinity of the final solution is crucial. Otherwise the method will converge to a non physical solution. A common good-enough voltage initial solution is to set all the voltages according to the slack voltage value.

7.3.1   Building the Jacobian matrix

The Jacobian matrix $[J]$ is the derivative of the fundamental power flow equation 7.1, with respect to the voltage magnitudes to be solved $[Vm]$ and $[\theta]$. It is one of the most expensive parts of the derivative based power flow methods.

In most literature, the reader will find the Jacobian presented with non matrix formulas which are hard to understand in their relation to the Jacobian matrix formation. Ray D. Zimmerman came up with a fantastic way of building the Jacobian matrix [1] by using lineal algebra operations over the complex matrices and vectors that represent the circuit. The formulation presents a huge advantage: No use of cosine and sine operations for the derivatives computation. The calculation of trigonometric functions is a recursive operation and should be avoided whenever possible in high performance computing. The formulation is presented as follows.

[1] Ray D Zimmerman. Ac power flows, generalized opf costs and their derivatives using complex matrix notation, 2010

First we compute the total current injection. It is a sparse diagonal matrix, so do not use a dense matrix format.

$$[I_{diag}] = diag([Y] \times [V] - [I]) \tag{7.17}$$

Then we convert the buses voltage solution $[V]$ into a square diagonal matrix. Use a sparse format for this.

$$[V_{diag}] = diag([V]) \tag{7.18}$$

Compute the buses voltage normalized solution $[V]$ into a square diagonal matrix. This is accomplished by dividing each complex voltage by its absolute value. Use a sparse format for this.

$$[E_{diag}] = diag\left(\frac{[V]}{[Vm]}\right) \tag{7.19}$$

Now, Compute the complex matrix derivative of the power injections $[S]$ with respect to the voltage module $[Vm]$:

$$\left[\frac{\partial S}{\partial Vm}\right] = [V_{diag}] \times \left([Y] \times [E_{diag}]\right)^* + [I_{diag}]^* \times [E_{diag}] \tag{7.20}$$

Compute the complex matrix derivative of the power injections $[S]$ with respect to the voltage angle $[\theta]$:

$$\left[\frac{\partial S}{\partial \theta}\right] = 1j \cdot [V_{diag}] \times \left([I_{diag}] - [Y] \times [V_{diag}]\right)^* \tag{7.21}$$

Finally, assemble the Jacobian matrix J. The Jacobian matrix is sparse and only contains real values. If the circuit contains npq buses of type PQ and npv buses of type PV, the Jacobian matrix is a square matrix with $2npq + npv$ rows and the same number of columns. See the figure 7.2.

$pqpv$ is a vector that contains the indices of the PQ and PV type indices in sequential order.

pq is a vector that contains the indices of the PQ type indices in sequential order.

$$J = \begin{bmatrix} Re\left\{\left[\frac{\partial S}{\partial \theta}\right]\right\}_{(pqpv,pqpv)} & Re\left\{\left[\frac{\partial S}{\partial Vm}\right]\right\}_{(pqpv,pq)} \\ Im\left\{\left[\frac{\partial S}{\partial \theta}\right]\right\}_{(pq,pqpv)} & Im\left\{\left[\frac{\partial S}{\partial Vm}\right]\right\}_{(pq,pq)} \end{bmatrix} \tag{7.22}$$

Observe that the Jacobian matrix is composed of four subsets of the previously computed derivatives $\left[\frac{\partial S}{\partial \theta}\right]$ and $\left[\frac{\partial S}{\partial Vm}\right]$, where combinations of the PQ and PV node indices are selected.

7.3.2   Newton-Raphson in power equations

Newton-Raphson is a recursive, iterative numerical method that minimizes the value of a function. In our case the value to minimize is the power mismatch:

$$[\Delta S] = [S_{specified}] - [S_{calc}] \tag{7.23}$$

Where:

$$[S_{calc}] = [V] \cdot ([Y] \times [V] - [I_{specified}])^* \tag{7.24}$$

The minimization of $[\Delta S]$ occurs via se recursive solution of the following linear system:

$$[J] \times [\Delta x] = [F] \tag{7.25}$$

Since the mismatch ($[\Delta S]$) is a vector of complex values, we transform it into a vector ($[F]$) with the real part of $[\Delta S]$ for the PQ and PV bus indices and the imaginary part of $[\Delta S]$ for the PQ bus indices of the vector. Observe that the length of the vector is $2npq + npv$, matching the Jacobian dimensions.

$$F = \begin{bmatrix} Re\{[\Delta S]_{(pvpq)}\} \\ Im\{[\Delta S]_{(pq)}\} \end{bmatrix} \tag{7.26}$$

Then we define the error to minimize as the infinite norm of F. The infinite norm is the same as the maximum absolute value. If the error is less than the specified tolerance, we stop the iterations otherwise, continue until a certain number of iterations considered as the maximum (for example 20).

$$error = ||F||_\infty = max(abs(F)) \tag{7.27}$$

An alternative version of the error is the equation 7.28, which is common in optimization problems.

The error expression from the equation 7.27 is commonly adopted for power flow problems since it is computationally less expensive than the equation 7.28.

$$error = \frac{1}{2} \cdot (F \times F^\top) \tag{7.28}$$

If the error is higher than the tolerance, we need to solve the voltages increment (Δx), for which we must compute the Jacobian first. Some simplifications of the method only compute the Jacobian one time, but for better accuracy, the Jacobian needs to be calculated on every iteration.

Do not invert $[J]$, instead use a linear system solver, to solve the linear system $J \times \Delta x = F$.

$$[\Delta x] = [J]^{-1} \times [F] \tag{7.29}$$

Notice that $[\Delta x]$ is a vector with the voltage angles $[\theta]$ for the PQ and PV nodes followed by the voltage modules $[Vm]$ for the PQ nodes. This matches the way we built the Jacobian $[J]$ and the mismatch $[F]$. Now

we must assign those polar voltage angle and module increments to the rectangular voltage vector that we use to compute the Jacobian and the mismatch.

To accomplish this, we declare two polar voltage increment vectors $[\Delta Vm]$ and $[\Delta\theta]$, initialize them to zero, and we fill the corresponding values with the following:

Here we use the auxiliary vectors $pqpv$ and pq which contain the indices of the pq and pv type buses, and the pq type buses respectively. $npqpv$ and npq are the sizes of those two vectors.

$$[\Delta\theta]_{(pqpv_i)} = [\Delta x]_i \quad \forall i \in 0..npqpv - 1 \tag{7.30}$$

$$\Delta[Vm]_{(pq_i)} = [\Delta x]_{(i+npqpv)} \quad \forall i \in 0..npq - 1 \tag{7.31}$$

Then we need to add those values to the voltage module and angle vectors $[Vm]$ and $[\theta]$:

μ is the iteration control parameter. Initially at each iteration $\mu = 1$.

$$[Vm]^{(k+1} = [Vm]^{(k} + \mu \cdot [\Delta Vm]^{(k} \tag{7.32}$$

$$[\theta]^{(k+1} = [\theta]^{(k} + \mu \cdot [\Delta\theta]^{(k} \tag{7.33}$$

At last, we convert the polar vectors $[Vm]$ and $[\theta]$ into a single complex voltage vector $[V]$.

Here we introduce the notion of the k_{th} iteration which represents the current iteration and the $k_{th} + 1$ represents the next iteration.

$$[V]^{(k+1} = [Vm]^{(k+1} \cdot \left(cos([\theta]^{(k+1}) + 1j \cdot sin([\theta]^{(k+1}) \right) \tag{7.34}$$

The method consists in repeating formulas 7.24 to 7.34 until the error is less or equal to the tolerance or a number of iterations is reached.

The algorithm is then:

1. Start.

2. Compute the power mismatch vector $[F]$ using the initial voltage solution $[V]$. Equation 7.26.

3. Compute the error. Equation 7.27.

4. While $error > tolerance$ or $iterations < max\_iterations$:

   (a) Compute the Jacobian

   (b) Solve the linear system. Equation 7.29.

   (c) Set μ: $\mu = 1$.

   (d) Assign $[\Delta x]$ to $[V]$. Equations 7.30 to 7.34.

   (e) Compute the power mismatch vector $[F]$ using the latest voltage solution $[V]$. Equation 7.26.

   (f) Compute the error. Equation 7.27.

   (g) If the error > error from the previous iteration:

      i. Decrease μ: $\mu = 0.25 \cdot \mu$

ii. Assign $[\Delta x]$ to $[V]$. Equations 7.30 to 7.34.

iii. Compute the power mismatch vector $[F]$ using the latest voltage solution $[V]$. Equation 7.26.

iv. Compute the error. Equation 7.27.

(h) $iterations = iterations + 1$

5. End.

The Newton-Raphson method tends to diverge if the grid is not perfectly balanced in loading and well conditioned (The impedances are not wildly different in per unit and X>R). The control parameter μ turns the Newton-Raphson method into a more controlled method that converges in most situations.

Here we have presented a rather simple way of determining the value of μ by decreasing it by four at every corrective step. Iwamoto and Tamura proposed in 1981 a method [2] to compute an optimal value of μ. This method involves the computation of the Jacobian matrix using ΔV instead of V.

[2] S Iwamoto and Y Tamura. A load flow calculation method for ill-conditioned power systems. *IEEE transactions on power apparatus and systems*, (4):1736–1743, 1981

7.3.3   *Newton-Raphson in current equations*

Newton-Raphson in current equations is similar to the regular Newton-Raphson algorithm presented before, but the mismatch is computed with the current instead of power.

The Jacobian is then:

$$J = \begin{bmatrix} Re\left\{\left[\frac{\partial I}{\partial \theta}\right]\right\}_{(pqpv,pqpv)} & Re\left\{\left[\frac{\partial I}{\partial Vm}\right]\right\}_{(pqpv,pq)} \\ Im\left\{\left[\frac{\partial I}{\partial \theta}\right]\right\}_{(pq,pqpv)} & Im\left\{\left[\frac{\partial I}{\partial Vm}\right]\right\}_{(pq,pq)} \end{bmatrix} \tag{7.35}$$

Where:

$$\left[\frac{\partial I}{\partial Vm}\right] = [Y] \times [E_{diag}] \tag{7.36}$$

$$\left[\frac{\partial I}{\partial \theta}\right] = 1j \cdot [Y] \times [V_{diag}] \tag{7.37}$$

The mismatch is computed as increments of current:

$$F = \begin{bmatrix} Re\{\Delta I\} \\ Im\{\Delta I\} \end{bmatrix} \tag{7.38}$$

Where:

$$[\Delta I] = \left(\frac{S_{specified}}{V}\right)^{*} - ([Y] \times [V] - [I^{specified}]) \tag{7.39}$$

The steps of the algorithm are equal to the the algorithm presented in 7.3.2

7.3.4   *Levenberg-Marquardt*

The Levenbarg-Marquardt is a recursive and iterative technique that is usually not related to power flow, but rather to non-linear least squares problems. Nevertheless, it solves the exact same problem as Newton-Raphson, but in a much more robust manner. It is advised when Newton-Raphson does not converge, because it exhibits excellent convergence properties at the cost of a higher computational effort.

At each iteration we need to assemble the system matrix $[A]$ as follows:

$$[A] = [J]^\top \times [J] + \lambda \cdot [Idn] \tag{7.40}$$

Where λ is computed only in the first iteration as:

$[Idn]$ is the identity matrix of compatible size. λ is a number. $[J]$ is the Jacobian matrix.

$$\lambda = 0.001 \cdot max(diag([J] \times [J]^\top)) \tag{7.41}$$

The right hand side of the linear system to obtain the voltage increments is:

F is provided by the equation 7.26.

$$[rhs] = [J]^\top \times [F] \tag{7.42}$$

Solve the voltage increments vector $[\Delta x]$. As in Newton-Raphson, this vector has $2npq + npv$ elements and the structure is the same as the one explained in the Newton-Raphson method. See figure 7.2.

Do not invert $[A]$, instead use a linear system solver, to solve the linear system $[A] \times [\Delta x] = [rhs]$.

$$[\Delta x] = [A]^{-1} \times [rhs] \tag{7.43}$$

Compute the objective function to minimize f. It is a value.

$$f = \frac{1}{2} \cdot \left([F] \times [F]^\top \right) \tag{7.44}$$

Calculate the decision function ρ. It is a value.

$f^{(k-1)}$ is the value of f in the previous iteration. $f^{(k}$ is the value of f computed in the current iteration. The value of $f^{(k-1}$ in the first iteration should be a very large number i.e. 1×10^9.

$$\rho = \frac{f^{(k-1} - f^{(k}}{\frac{1}{2} \cdot \left([\Delta x] \times (\lambda \cdot [\Delta x] + [rhs])^\top \right)} \tag{7.45}$$

Now, based on the value of ρ we decide what to do in the next iteration.

If ρ is greater than zero, we mark a flag to update the Jacobian in the next iteration, and we update the voltage solution using Δx as described in equations 7.30 to 7.34. We also need to modify λ:

$$\lambda = \lambda \cdot max\left(\frac{1}{3}, 1 - (2 \cdot \rho - 1)^3\right) \tag{7.46}$$

And set a variable $\nu = 2$.

If ρ is less or equal to zero, then the values of Δx will not improve the solution and we need to take corrective actions. We mark a flag to not to update the Jacobian in the next iteration, we set $\lambda = \lambda \cdot \nu$ and $\nu = \nu \cdot 2$.

The complete algorithm is:

1. Start.

2. Compute the mismatch power vector $[F]$ using the initial voltage solution $[V]$. Equation 7.26.

3. Compute the error. Equation 7.27.

4. While $error > tolerance$ or $iterations < max\_iterations$:

   (a) If the computation of the Jacobian is enabled, compute the Jacobian and the system matrix $[A]$ using equation 7.40.

   (b) Compute the linear system right hand side using equation 7.42.

   (c) Solve the linear system to obtain $[\Delta x]$. Equation 7.43.

   (d) Compute the objective function value f using equation 7.44.

   (e) Compute the decision value ρ using equation 7.45.

   (f) If $\rho > 0$:

       i. Assign $[\Delta x]$ to $[V]$. Equations 7.30 to 7.34.
       ii. Update λ using equation 7.46.
       iii. Set $v = 2$.
       iv. Set the Jacobian update flag to true.

   (g) If $\rho \leq 0$:

       i. Assign $[\Delta x]$ to $[V]$. Equations 7.30 to 7.34.
       ii. Update $\lambda = v \cdot \lambda$.
       iii. Update $v = 2 \cdot v$.
       iv. Set the Jacobian update flag to false.

   (h) Compute the mismatch function $[F]$ using the latest voltage solution $[V]$. Equation 7.26.

   (i) Compute the error. Equation 7.27.

   (j) $iterations = iterations + 1$

5. End.

7.4 Linear power flow

The so called direct current power flow (or just DC power flow) is a convenient oversimplification of the power flow procedure.

It assumes that in any branch the reactive part of the impedance is much larger than the resistive part, hence the resistive part is neglected, and that all the voltages modules are the nominal per unit values. This is, $[Vm] = [1]$ for load nodes and $[Vm] = [V_{set}]$ for the generator nodes, where $[V_{set}]$ is the generator set point value.

In order to compute the DC approximation we must perform a transformation. The slack nodes are removed from the grid, and their influence is maintained by introducing equivalent currents in all the nodes. The equivalent admittance matrix $[Y_{red}]$ is obtained by removing the rows and columns corresponding to the slack nodes. Likewise the removed elements conform the $[Y_{slack}]$ matrix.
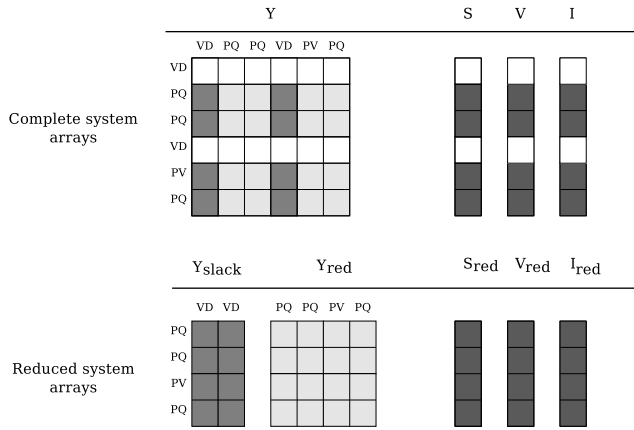
$$[P_{red}] = \left[P_{red}^{specified}\right] + (-[B_{slack}] \times [\theta_{slack}] + Re\{[I_{red}]\}) \cdot [|V_{red}|] \quad (7.47)$$

The equation 7.47 computes the DC power injections as the sum of the different factors mentioned:

1. $[P_{red}^{specified}]$: Real part of the reduced power injections.

2. $[B_{slack}] \times [\theta_{slack}] \cdot [Vm_{red}]$: Currents that appear by removing the slack nodes while keeping their influence, multiplied by the voltage module to obtain power.

3. $Re\{[I_{red}]\} \cdot [Vm_{red}]$: Real part of the grid reduced current injections, multiplied by the voltage module to obtain power.

Where:
$$\left[P_{red}^{specified}\right] = Re\left\{\left[S_{red}^{specified}\right]\right\}$$
$$[B_{slack}] = Im\{[Y_{slack}]\}$$
$$[\theta_{slack}] = angle([V_{slack}])$$

Once the power injections are computed, the new voltage angles are obtained by:

$$[\theta_{red}] = [B_{red}]^{-1} \times [P_{red}] \quad (7.48)$$

The new voltage is then:

$$[V_{red}] = [Vm_{red}] \cdot e^{1j \cdot [\theta_{red}]} \quad (7.49)$$

Where:
$$[B_{red}] = Im\{[Y_{red}]\}$$

This method usually produces a solution with a large power mismatch. That is to be expected because the method is an oversimplification with no iterative convergence criteria, just a straight forward set of operations.

7.5   A better linear power flow

The linear approximation presented in this section is a much more convenient linearisation of the power flow equation, because it provides the voltage module and angle. This procedure is great for real time applications such as large grids SCADA applications.

Using the formulation presented in [3], we obtain a way to solve circuits in

[3] Priscila Rossoni, William Moreti da Rosa, and Edmarcio Antonio Belati. Linearized ac load flow applied to analysis in electric power systems. *IEEE Latin America Transactions*, 14(9):4048–4053, 2016

one shot (without iterations) with quite positive results for a linear approximation.

$$
\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} \Delta\theta \\ \Delta Vm \end{bmatrix} = \begin{bmatrix} Rhs_1 \\ Rhs_2 \end{bmatrix}
\tag{7.50}
$$

Where:

- $A_{11} = -Im\{[Y_{series}]_{(pqpv, pqpv)}\}$

- $A_{12} = Re\{[Y_{bus}]_{(pqpv, pq)}\}$

- $A_{21} = -Im\{[Y_{series}]_{(pq, pqpv)}\}$

- $A_{22} = -Re\{[Y_{bus}]_{(pq, pq)}\}$

- $Rhs_1 = Re\{[S]_{(pqpv)}\}$

- $Rhs_2 = Im\{[S]_{(pq)}\}$

Here, $[Y_{bus}]$ is the normal circuit admittance matrix and $[Y_{series}]$ is the admittance matrix formed with only series elements of the π model, this is, neglecting all the shunt admittances.

Solving the vector $[\Delta\theta + 0, \Delta Vm + 1]$ we get θ for the pq and pv nodes and Vm for the pq nodes.

The solution obtained has a lower mismatch than the DC approximation usually in the order of 0.1 p.u. which is acceptable for a linear method. The error is lower because the voltage module is also computed.

7.6   Holomorphic embedding

First introduced by Antonio Trias in 2012 [4], promises to be a non-divergent power flow method. Trias originally developed a version with no voltage controlled nodes (PV), in which the convergence properties are excellent.

The version programmed in the file HELM.py has been adapted from the master thesis of Muthu Kumar Subramanian at the Arizona State University (ASU) [5]. This version includes a formulation of the voltage controlled nodes. My experience indicates that the introduction of the PV control deteriorates the convergence properties of the holomorphic embedding method. However, in many cases, it is the best approximation to a solution. especially when Newton-Raphson does not provide one.

7.6.1   Concepts

All the power flow algorithms until the HELM method was introduced were iterative and recursive. The helm method is iterative but not recursive. A simple way to think of this is that traditional power flow methods are exploratory, while the HELM method is a planned journey. In theory the HELM method is superior, but in practice the numerical degeneration makes it less ideal.

The fundamental idea of the recursive algorithms is that given a voltage initial point (1 p.u. at every node, usually) the algorithm explores the surroundings of the initial point until a suitable voltage solution is reached or no

[4] A. Trias. The holomorphic embedding load flow method. pages 1–8, July 2012. ISSN 1944-9925. DOI: 10.1109/PESGM.2012.6344759

[5] Muthu Kumar Subramanian. Application of holomorphic embedding to the power-flow problem, 2014

solution at all is found because the initial point is supposed to be "far" from the solution.

On the HELM methods, we form a "curve" that departures from a known mathematically exact solution that is obtained from solving the grid with no power injections. This is possible because with no power injections, the grid equations become linear and straight forward to solve. The arriving point of the "curve" is the solution that we want to achieve. That "curve" is best approximated by a Padé approximation. To compute the Padé approximation we need to compute the coefficients of the unknown variables, in our case the voltages (and possibly the reactive powers at the PV nodes).

The HELM formulation consists in the derivation of formulas that enable the calculation of the coefficients of the series that describes the "curve" from the mathematically know solution to the unknown solution. Once the coefficients are obtained, the Padé approximation computes the voltage solution at the "end of the curve", providing the desired voltage solution. The more coefficients we compute the more exact the solution is (this is true until the numerical precision limit is reached).

All this sounds very strange, but it works ;)

If you want to get familiar with this concept, you should read about the homotopy concept. In practice the continuation power flow does the same as the HELM algorithm, it takes a known solution and changes the loading factors until a solution for another state is reached.

7.6.2   Fundamentals

The fundamental equation that defines the power flow problem is:

$$S = V \times (Y \times V)^* \tag{7.51}$$

Most usefully represented like this:

$$Y \times V = \left(\frac{S}{V}\right)^* \tag{7.52}$$

The holomorphic embedding is to insert a "travelling" parameter α, such that for $\alpha = 0$ we have an mathematically exact solution of the problem (but not the solution we're looking for...), and for $\alpha = 1$ we have the solution we're looking for. The other thing to do is to represent the variables to be computed as McLaurin series. Let's go step by step.

For $\alpha = 0$ we say that $S = 0$, in this way the equation 7.52 becomes linear, and its solution is mathematically exact. But for that to be useful in our case we need to split the admittance matrix Y into Y_{series} and Y_{shunt}. Y_{shunt} is a diagonal matrix, so it can be expressed as a vector instead (no need for matrix-vector product).

$$Y_{series} \times V = \left(\frac{S}{V}\right)^* - Y_{shunt} V \tag{7.53}$$

This is what will allow us to find the zero "state" in the holomorphic series calculation. For $\alpha = 1$ we say that $S = S$, so we don't know the voltage

solution, however we can determine a path to get there:

$$Y \times V(\alpha) = \left(\frac{\alpha S}{V(\alpha)} \right)^* - \alpha Y_{shunt} \times V(\alpha) = \frac{\alpha S^*}{V(\alpha)^*} - \alpha Y_{shunt} V(\alpha)$$

(7.54)

Wait, what?? did you just made this stuff up??, well so far my reasoning is:

- The voltage V is what I have to convert into a series, and the series depend of α, so it makes sense to say that V, as it is dependent of α, becomes $V(\alpha)$.

- Regarding the α that multiplies S, the amount of power (αS) is what I vary during the *travel* from $\alpha = 0$ to $\alpha = 1$, so that is why S has to be accompanied by the *traveling* parameter α.

- In my opinion the $\alpha \, Y_{shunt}$ is to provoke the first voltage coefficients to be one. $Y_{series} \times V[0] = 0$, makes $V[0] = 1$. This is essential for later steps (is a condition to be able to use Padé).

The series are expressed as McLaurin equations:

$$V(\alpha) = \sum_{n}^{\infty} V_n \alpha^n$$

(7.55)

7.6.3 *Holomorphicity check*

There's still something to do. The magnitude $(V(\alpha))^*$ has to be converted into $(V(\alpha^*))^*$. This is done in order to make the function be holomorphic. The holomorphicity condition is tested by the Cauchy-Riemann condition, this is $\partial V / \partial \alpha^* = 0$, let's check that:

$$\partial \left(V(\alpha)^* \right) / \partial \alpha^* = \partial \left(\sum_{n}^{\infty} V_n^* (\alpha^n)^* \right) / \partial \alpha^* = \sum_{n}^{\infty} \alpha^n V_n^* (\alpha^{n-1})^* \quad (7.56)$$

Which is not zero, obviously. Now with the proposed change:

$$\partial \left(V(\alpha^*) \right)^* / \partial \alpha^* = \partial \left(\sum_{n}^{\infty} V_n^* \alpha^n \right) / \partial \alpha^* = 0$$

(7.57)

Yay!, now we're mathematically happy, since this stuff has no effect in practice because our α is not going to be a complex parameter, but for sake of being correct the equation is now:

$$Y_{series} \times V(\alpha) = \frac{\alpha S^*}{V^*(\alpha^*)} - \alpha Y_{shunt} V(\alpha)$$

(7.58)

The fact that $V^*(\alpha^*)$ is dividing is problematic. We need to express it as its inverse so it multiplies instead of divide.

$$\frac{1}{V(\alpha)} = W(\alpha) \longrightarrow W(\alpha)V(\alpha) = 1 \longrightarrow \sum_{n=0}^{\infty} W_n \alpha^n \sum_{n=0}^{\infty} V_n \alpha^n = 1 \;\; (7.59)$$

Expanding the series and identifying terms of α we obtain the expression to compute the inverse voltage series coefficients:

$$W_n = \begin{cases} \dfrac{1}{V_0}, & n = 0 \\[2ex] -\dfrac{\displaystyle\sum_{k=0}^{n} W_k V_{n-k}}{V_0}, & n > 0 \end{cases} \tag{7.60}$$

Now, the equation 7.58 is:

$$Y_{series} \times V(\alpha) = \alpha S^* \cdot W(\alpha)^* - \alpha Y_{shunt} V(\alpha) \tag{7.61}$$

Substituting the series by their McLaurin expressions:

$$Y_{series} \times \sum_{n=0}^{\infty} V_n \alpha^n = \alpha S^* \left(\sum_{n=0}^{\infty} W_n \alpha^n \right)^* - \alpha Y_{shunt} \sum_{n=0}^{\infty} V_n \alpha^n \tag{7.62}$$

Expanding the series an identifying terms of α we obtain the expression for the voltage coefficients:

$$V_n = \begin{cases} 0, & n = 0 \\ S^* W_{n-1}^* - Y_{shunt} V_{n-1}, & n > 0 \end{cases} \tag{7.63}$$

This is the HELM fundamental formula derivation for a grid with no voltage controlled nodes (no PV nodes). Once a sufficient number of coefficients are obtained, we still need to use the Padé approximation to get voltage values out of the series.

In the previous formulas, the number of the bus has not been explicitly detailed. All the V and the W are matrices of dimension $n \times nbus$ (number of coefficients by number of buses in the grid) This structures are depicted in the figure 7.4. For instance V_n is the nth row of the coefficients structure V.
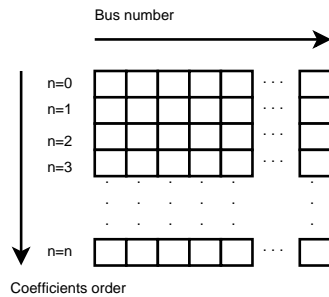
Figure 7.4: Structure of the coefficients

7.6.4   *Padé approximation*

The equation 7.55 provides us with an expression to obtain the voltage from the coefficients, knowing that for $\alpha = 1$ we get the final voltage results. So, why do we need any further operation?, and what is this Padé thing?

Well, it is true that the equation 7.55 provides an approximation of the voltage by means of a series (this is similar to a Taylor approximation), but in

practice, the approximation might provide a wrong value for a given number of coefficients. The Padé approximation accelerates the convergence of any given series, so that you get a more accurate result with less coefficients. This means that for the same series of voltage coefficients, using the equation 7.55 could give a completely wrong result, whereas by applying Padé to those coefficients one could obtain a fairly accurate result.

The Padé approximation is a rational approximation of a function. In our case the function is $V(\alpha)$, represented by the coefficients structure V. The approximation is valid over a small domain of the function, in our case the domain is $\alpha = [0, 1]$. The method requires the function to be continuous and differentiable for $\alpha = 0$. Hence the Cauchy-Riemann condition. And yes, our function meets this condition, we tested it before.

Padé approximation algorithm   The canonical Padé algorithm for our problem is described by:

$$Voltage\_value\_approximation = \frac{P_N(\alpha)}{Q_M(\alpha)} \quad \forall \alpha \in [0, 1] \qquad (7.64)$$

Here $N = M = n/2$, where n is the number of available voltage coefficients, which has to be an even number to be exactly divisible by 2. P and Q are polynomials which coefficients p_i and q_i must be computed. It turns out that if we make the first term of $Q_M(\alpha)$ be $q_0 = 1$, the function to be approximated is given by the McLaurin expression (What a happy coincidence!)

$$P_N(\alpha) = p_0 + p_1\alpha + p_2\alpha^2 + ... + p_N\alpha^N \qquad (7.65)$$

$$Q_M(\alpha) = 1 + q_1\alpha + q_2\alpha^2 + ... + q_M\alpha^M \qquad (7.66)$$

The problem now boils down to find the coefficients q_i and p_i. This is done by solving two systems of equations. The first one to find q_i which does not depend on p_i, and the second one to get p_i which does depend on q_i.

First linear system: The only unknowns are the q_i coefficients.

$$
\begin{aligned}
q_M V_{N-M+1} + q_{M-1} V_{N-M+2} + ... + q_1 V_N &= 0 \\
q_M V_{N-M+2} + q_{M-1} V_{N-M+3} + ... + q_1 V_{N+1} &= 0 \\
... & \\
q_M V_N + q_{M-1} V_{N+1} + ... + q_1 V_{N+M+1} + V_{N+M} &= 0
\end{aligned}
\qquad (7.67)
$$

Second linear System: The only unknowns are the p_i coefficients.

$$
\begin{aligned}
V_0 - p_0 &= 0 \\
q_1 V_0 + V_1 p_1 &= 0 \\
q_2 V_0 + q_1 V_1 + V_2 - p_2 &= 0 \\
q_3 V_0 + q_2 V_1 + q_1 V_2 + V_3 - p_3 &= 0 \\
... & \\
q_M V_{N-M} + q_{M-1} V_{N-M+1} + ... + + V_N - p_N &= 0
\end{aligned}
\qquad (7.68)
$$

Once the coefficients are there, you would have defined completely the polynomials $P_N(\alpha)$ and $Q_M(\alpha)$, and it is only a matter of evaluating the equation 7.64 for $\alpha = 1$.

This process is done for every column of coefficients $V = \{V_0, V_1, V_2, V_3, ..., V_n\}$ of the structure depicted in the figure 7.4. This means that we have to perform a Padé approximation for every node, using the one columns of the voltage coefficients per Padé approximation.

Wynn's Padé approximation algorithm   Wynn published a paper in 1969 where he proposed a simple calculation method to obtain the Padé approximation. This method is based on a table. Weniger in 1989 publishes his thesis where a faster version of Wynn's algorithm is provided in Fortran code.

One of the advantages of this method over the canonical Padé approximation implementation is that it can be used for every iteration. In the beginning I thought it would be faster but it turns out that it is not faster since the amount of computation increases with the number of coefficients, whereas with the canonical implementation the order of the matrices does not grow dramatically and it is executed the half of the times.

On top of that my experience shows that the canonical implementation provides a more consistent convergence.

Anyway, both implementations are there to be used in the code.

7.6.5   *Formulation with PV nodes*

The section 7.6.2 introduces the canonical HELM algorithm. That algorithm does not include the formulation of PV nodes. Other articles published on the subject feature PV formulations that work more or less to some degree. The formulation below is a formulation corrected by myself from a formulation contained here [6], which does not work as published, hence the correction.

[6] Chengxi Liu, Bin Wang, Fengkai Hu, Kai Sun, and Claus Leth Bak. Online voltage stability assessment for load areas based on the holomorphic embedding method. *IEEE Transactions on Power Systems*, 2017

Embedding   The following embedding equations are proposed instead of the canonical HELM equations from section 7.6.2.

For Slack nodes:

$$V(\alpha) = V^{SP} \quad \forall \alpha = 0 \tag{7.69}$$

For PQ nodes:

$$\begin{cases} Y \times V(\alpha) = 0 & \forall \alpha = 0 \\ Y \times V(\alpha) = \dfrac{\alpha S}{V^*(\alpha^*)} & \forall \alpha > 0 \end{cases} \tag{7.70}$$

For PV nodes:

$$\begin{cases} Y \times V(\alpha) = \dfrac{S}{V^*(\alpha^*)} & \forall \alpha = 0 \\ Y \times V(\alpha) = \dfrac{S - jQ(\alpha)}{V^*(\alpha^*)} & \forall \alpha > 0 \end{cases} \tag{7.71}$$

$$\begin{cases} V(\alpha)V^*(\alpha^*) = |V_0|^2 & \forall \alpha = 0 \\ V(\alpha)V^*(\alpha^*) = |V_0|^2 + (|V^{SP}|^2 - |V_0|^2) & \forall \alpha > 0 \end{cases} \tag{7.72}$$

This embedding translates into the following formulation:

Step 1   The formulas are adapted to exemplify a 3-bus system where
the bus1 is a slack, the bus 2 is PV and the bus 3 is PQ. This follows the
example of the Appendix A of [7].

[7] Chengxi Liu, Bin Wang, Fengkai Hu, Kai Sun, and Claus Leth Bak. Online voltage stability assessment for load areas based on the holomorphic embedding method. *IEEE Transactions on Power Systems*, 2017

Compute the initial no-load solution ($n = 0$):

$$
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
G_{21} & -B_{21} & G_{22} & -B_{22} & G_{23} & -B_{23} \\
B_{21} & G_{21} & B_{22} & G_{22} & B_{23} & G_{23} \\
G_{31} & -B_{31} & G_{32} & -B_{32} & G_{33} & -B_{33} \\
B_{31} & G_{31} & B_{32} & G_{32} & B_{33} & G_{33}
\end{bmatrix}
\times
\begin{bmatrix}
V[n]_{re,1} \\
V[n]_{im,1} \\
V[n]_{re,2} \\
V[n]_{im,2} \\
V[n]_{re,3} \\
V[n]_{im,3}
\end{bmatrix}
=
\begin{bmatrix}
V_{re,1}^{SP} \\
V_{im,1}^{SP} \\
0 \\
0 \\
0 \\
0
\end{bmatrix}
\quad \forall n = 0
$$
(7.73)

Form the solution vector $V[n]$ you can compute the buses calculated
power and then get the reactive power at the PV nodes to initialize $Q[0]$:

$$
S = V[0] \cdot (Y_{bus} \times V[0])^* \tag{7.74}
$$

$$
Q_i[0] = Im\{S_i\} \quad \forall i \in PV \tag{7.75}
$$

The initial inverse voltage coefficients $W[0]$ are obtained by:

$$
W_i[0] = \frac{1}{V_i[0]} \quad \forall i \in N \tag{7.76}
$$

This step is entirely equivalent to find the no load solution using the
Z-Matrix reduction.

Step 2   Construct the system of equations to solve the coefficients of order
greater than zero ($n > 0$). Note that the matrix is the same as constructed
for the previous step, but adding a column and a row for each PV node to
account for the reactive power coefficients. In our 3-bus example, there is
only one PV node, so we add only one column and one row.

$$
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 \\
G_{21} & -B_{21} & G_{22} & -B_{22} & G_{23} & -B_{23} & W[0]_{im} \\
B_{21} & G_{21} & B_{22} & G_{22} & B_{23} & G_{23} & W[0]_{re} \\
G_{31} & -B_{31} & G_{32} & -B_{32} & G_{33} & -B_{33} & 0 \\
B_{31} & G_{31} & B_{32} & G_{32} & B_{33} & G_{33} & 0 \\
0 & 0 & V[0]_{re} & V[0]_{im} & 0 & 0 & 0
\end{bmatrix}
\times
\begin{bmatrix}
V[n]_{re,1} \\
V[n]_{im,1} \\
V[n]_{re,2} \\
V[n]_{im,2} \\
V[n]_{re,3} \\
V[n]_{im,3} \\
Q_2[n]
\end{bmatrix}
=
\begin{bmatrix}
0 \\
0 \\
f2_{re} \\
f2_{im} \\
f1_{re} \\
f1_{im} \\
\epsilon[n]
\end{bmatrix}
\quad \forall n > 0
$$
(7.77)

Where:

$$f1 = S_i^* \cdot W_i^*[n-1] \quad \forall i \in PQ \tag{7.78}$$

$$f2 = P_i \cdot W_i^*[n-1] + conv(n, Q_i, W_i^*) \quad \forall i \in PV \tag{7.79}$$

$$\epsilon[n] = \delta_{n1} \cdot \frac{1}{2} \left(|V_i^S P|^2 - |V_i[0]|^2 \right) - \frac{1}{2} conv(n, V_i, V_i^*) \quad \forall i \in PV, n > 0 \tag{7.80}$$

δ_{nk} is a function that returns 1 when $n == k$ and 0 otherwise.

The convolution $conv$ is defined as:

$$conv(n, A, B) = \sum_{m=0}^{n-1} A[m] \cdot B[n-m] \tag{7.81}$$

The system matrix (A_{sys}) is the same for all the orders of $n > 0$, therefore we only build it once, and we factorize it to solve the subsequent coefficients.

After the voltages $V[n]$ and the reactive power at the PV nodes $Q[n]$ is obtained solving the linear system (eq 7.77), we must solve the inverse voltage coefficients of order n for all the buses:

$$W_i[n] = \frac{-\sum_{m=0}^{n} W_i[m] \cdot V_i[n-m]}{V_i[0]} \quad \forall i \in N, n > 0 \tag{7.82}$$

Step 3   Repeat step 2 until a sufficiently low error is achieved or a maximum number of iterations (coefficients).

The error is computed by comparing the calculated power S (eq 7.74) with the specified power injections S^{SP}:

$$mismatch = S - S^{SP} \tag{7.83}$$

$$error = |mismatch|_\infty = max(abs(mismatch)) \tag{7.84}$$

7.7   Post voltage solution: Compute the power flows

Ironically, what is called the power flow problem does not compute the power flows in the branches of a grid. It computes the node voltages. In this section we explain how to compute the current and power that flows through the grid branches using a given voltage solution vector.

First we compute the branches per unit currents in the *from* and *to* sides:

$$[I_f] = ([C_f] \times [Y_{bus}]) \times [V] \qquad (7.85)$$

$$[I_t] = ([C_t] \times [Y_{bus}]) \times [V] \qquad (7.86)$$

These are matrix-vector multiplications. The result is the per unit currents vector flowing through a branch, seen from the *from* bus or from the *to* bus.

Then we compute the power values in the *from* and *to* sides:

$$[S_f] = ([C_f] \times [V]) \cdot [I_f]^* \qquad (7.87)$$

$$[S_t] = ([C_t] \times [V]) \cdot [I_t]^* \qquad (7.88)$$

These are element-wise multiplications of two vectors, resulting in the per unit power flowing through a branch seen from the *from* bus or from the *to* bus.

Now we can compute the losses in MVA as:

$$losses = |[S_f] - [S_t]| \cdot Sbase \qquad (7.89)$$

And also the branches loading in per unit as:

$$loading = \frac{max(|[S_f]|, |[S_t]|) \cdot Sbase}{[rate]} \qquad (7.90)$$

Another way of computing the loading would be to perform the average of the injected powers:

$$loading = \frac{max(|[S_f] - [S_t]|) \cdot Sbase}{[rate]} \qquad (7.91)$$

Finally we compute the power in all nodes:

$$[S_{bus}] = [V] \cdot ([Y_{bus}] \times [V] - [I_{bus}])^* \qquad (7.92)$$

The imaginary part of $[S_{bus}]$ at the PV nodes is he calculated reactive power.

$$[Q_{pv}] = Im\{[S_{bus}]_{(pv)}\} \qquad (7.93)$$

The variables are:

- $[Y_{bus}]$: Bus admittance matrix in p.u. See equation 6.17.
- $[C_f]$: Connectivity matrix of the branches and the *from* buses.
- $[C_t]$: Connectivity matrix of the branches and the *to* buses.
- $[V]$: Array of bus voltages in p.u.
- $[I_f]$: Array of currents at the *from* buses in p.u.
- $[I_t]$: Array of currents at the *to* buses in p.u.
- $[S_f]$: Array of powers at the *from* buses in p.u.
- $[S_t]$: Array of powers at the *to* buses in p.u.
- $[rate]$: Array of branch ratings in MVA.

Note that $[S_f]$ has the opposite sign of $[S_t]$, therefore to sum the two values we actually need to subtract them.

The calculated reactive power is used to check the generation reactive power limits after the power flow.

7.8   *Reactive power control: The outer loop*

The power flow result seen in the previous section 7.7 computes values of reactive power for the PV nodes $[Q_{pv}]$. These reactive power values may exceed the reactive power capabilities of the generators connected at those buses. The commonly accepted practice is to turn the PV buses where the reactive power exceeds the limits to PQ and re-run a power flow simulation. If this simulation is successful the Bus type is reverted to PV and retry.

Many academic texts suggest that the PV-PQ switching has to be done within the iteration of the power flow solution. That is fundamentally wrong because it introduces discontinuities in the power flow numerical process that is assumed to be smooth. Hence if the PQ-PV switching is done inside the power flow numerical loop, it will be a miracle if the process converges.

The PQ-PV switching logic formulated at [8] is the following:

1. Bus i is a PQ bus in the previous iteration and its reactive power was fixed at its lower limit:

    - If the bus voltage magnitude $Vm_i \geq Vset_i$, then it remains a PQ bus at current iteration and set $Q_i = Qmin_i$.
    - If $Vm_i < Vset_i$, then compare Q_i with the upper and lower limits.
        - If $Q_i \geq Qmax_i$, then it is still a PQ bus but set $Q_i = Qmax_i$.
        - If $Q_i \leq Qmin_i$, then it is still a PQ bus and set $Q_i = Qmin_i$.
        - If $Qmin_i < Q_i < Qmax_i$, then it is switched to PV bus, set $V_i = Vset_i \cdot e^{-j \cdot angle(V_i)}$.

2. Bus i is a PQ bus in the previous iteration and its reactive power was fixed at its upper limit:

    - If its voltage magnitude $Vm_i \leq Vset_i$, then the bus i still a PQ bus and set $Q_i = Qmax_i$.
    - If $Vm_i > Vset_i$, then compare between Q_i and its upper/lower limits:
        - If $Q_i \geq Qmax_i$, then it is still a PQ bus and set $Q_i = Qmax_i$.
        - If $Q_i \leq Qmin_i$, then it is still a PQ bus but let $Q_i = Qmin_i$ in current iteration.
        - If $Qmin_i < Q_i < Qmax_i$, then it is switched to PV bus and set $V_i = Vset_i \cdot e^{-j \cdot angle(V_i)}$

3. Bus i is a PV bus in the previous iteration.

    Compare Q_i with its upper and lower limits.

    - If $Q_i \geq Qmax_i$, then it is switched to PQ and set $Q_i = Qmax_i$.
    - If $Qi \leq Qmin_i$, then it is switched to PQ and set $Q_i = Qmin_i$.
    - If $Qmin_i < Q_i < Qmax_i$, then it is still a PV bus.

The switching logic is applied only to the buses that are PV in the first place. There is need to apply this to all the nodes. Because of this switching logic, there shall be only one voltage controlled device per node. Otherwise it is quite hard to ensure that all the machines connected to a particular bus fulfil their reactive power limits.

[8] Jinquan Zhao, Hsiao-Dong Chiang, Ping Ju, and Hua Li. On pv-pq bus type switching logic in power flow computation. In *Power Systems Computation Conference (PSCC), Glasgow, Scotland*, 2008
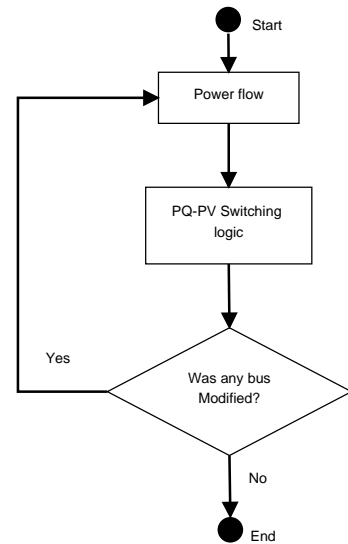
Figure 7.5: $PQ - PV$ switching algorithm.

8 Optimal power flow (OPF)

The optimal power flow are a set of problems which implementation call for the usage of MIP solvers (Mixed-Integer Programming solvers). However that is not a hard requirement and other methods might be used such as genetic algorithms, convex solvers, etc.

8.1 Optimal power flow

The power flow problem consists in finding the voltages of a circuit given the nodal power injections and the circuit admittances. The power flow equation in summation expression is:

$$S_i = V_i \sum_{j}^{nodes} \left(Y_{ij} \cdot V_j \right)^* \quad \forall i \in nodes \tag{8.1}$$

The non-vectorized expression is of use now, because the MIP formulation of the OPF problem does not allow vectorization usually.

The optimal power flow problem consists in finding the generators power injections that supply the demand and the losses while minimizing the grid violations (Voltages out of bounds and branch flows limits). Here we are going to adapt the linear power flow formulation developed in the section 7.4 in order to accommodate the generators dispatch.

Minimize the generators dispatch cost, including the cost of the slack generators:

$$min : \sum_{i}^{Generators+Slack} cost_i \cdot PG_i \tag{8.2}$$

ST: Generator limits must be within boundaries (include the slack generator as well)

$$PG_i^{min} \leq PG_i \leq PG_i^{max} \tag{8.3}$$

The voltages must satisfy the power balance in every PQ and PV node.

$$PG_i - PD_i = \sum_{j}^{PQPV} B_{ij} \cdot \theta_j \quad \forall i \in PQPV \tag{8.4}$$

When traversing the matrix B, try to avoid two for loops, dealing with B as if it was dense because it is not. Ideally B should be in CSC format. Then traversing B in sparse mode is several orders of magnitude faster.

The slack voltage angles are set to zero.

$$\theta_k = 0 \quad \forall k \in Slack \tag{8.5}$$

The branch flows must be within limits.

$$-F_{ij}^{max} \leq B_{ij}(\theta_i - \theta_j) \leq F_{ij}^{max} \qquad (8.6)$$

This is reformulated as:

$$B_{ij}(\theta_i - \theta_j) \leq F_{ij}^{max} \qquad (8.7)$$

$$B_{ij}(\theta_j - \theta_i) \leq F_{ij}^{max} \qquad (8.8)$$

Set the slack generator power:

$$PG_i - PD_i = \sum_{j}^{nodes} B_{ij} \cdot \theta_j \quad \forall i \in Slack \qquad (8.9)$$

- P_i: Active power injection at the node i.

- B: Susceptance matrix. It is the imaginary part of the circuit admittance matrix.

- θ: Voltage angles in radians.

- $PQPV$: Set of the non-slack nodes.

- $Slack$: Set of the slack nodes.

There are plenty of other formulations that feature ramps, multiple periods and the modelling of other devices. For an overview of such methods see [1].

[1] Joshua Adam Taylor. *Convex optimization of power systems*. Cambridge University Press, 2015

8.2 A better optimal power flow

This section introduces the use of a linear power flow derived from the power flow algorithm presented in the section 7.5. In this formulation, both the voltage module and angle are subject to control. In the power flow problem the power values are fixed whereas in the optimal power flow both generation and voltage are unknown.

The linear program is:

$$min: \sum_{i}^{Generators} cost_i \cdot PG_i \quad \forall i \in generators \qquad (8.10)$$

Subjected To:

- The generators generation power PG_i must be within limits.

$$PG_i^{min} \leq PG_i \leq PG_i^{max} \quad \forall i \in generators \qquad (8.11)$$

- The nodes voltage angle θ_i must be within limits. Wise limits are $\theta_i^{min} = 0$ and $\theta_i^{max} = 0.5 \quad Rad$.

$$\theta_i^{min} \leq \theta_i \leq \theta_i^{max} \quad \forall i \in pvpq \qquad (8.12)$$

- The nodes voltage module V_i must be within limits. Wise limits are $V_i^{min} = 0.9 \quad p.u.$ and $V_i^{max} = 1.1 \quad p.u.$

$$Vm_i^{min} \leq Vm_i \leq Vm_i^{max} \quad \forall i \in pq \qquad (8.13)$$

The equation 7.50 can be transformed to the following equations:

- Power balance in the modes: The power from the voltage solution must match exactly the specified power (in active and reactive components).

Remember that linear program libraries cannot deal with complex numbers, therefore we have to split the equations into real and imaginary parts. Also remember that there exist $Y = G + j \cdot B$ (complete admittance matrix) and $Y_S = G_S + j \cdot B_S$ (Admittance matrix of the series components).

$$-\sum_{k}^{pvpq} Bs_{k,i} \cdot \Delta\theta_i + \sum_{k}^{pq} G_{k,i} \cdot \Delta Vm_i = \sum_{g}^{generators_i} PG_g - \sum_{l}^{loads_i} PD_l \quad \forall i \in pvpq \qquad (8.14)$$

$$-\sum_{k}^{pqpv} Gs_{k,i} \cdot \Delta\theta_i - \sum_{k}^{pq} B_{k,i} \cdot \Delta Vm_i = -\sum_{l}^{loads_i} QD_l \quad \forall i \in pq \qquad (8.15)$$

After solving the voltage deltas ($\Delta\theta$ and ΔV) with the linear program, we can get the actual voltage angles and module with the following equations.

$$\theta_i = \Delta\theta_i \qquad (8.16)$$

$$Vm_i = Vm_i^{specified} + \Delta Vm_i \qquad (8.17)$$

9 Time series power flow

The time series power flow is nothing more than power flows executed sequentially for series of profiles of load and generation. The usage of this simulation is to inspect how does the voltage and loading behave given a profile of the load and generation of the grid.

Algorithm

1. Compile the load and generation profiles

2. initialize the control variables. $t = 0$.

3. for each profile step:

   - Pick grid load state at t, and for the injection vectors S and I.

   - Run a power flow simulation with those injection vectors and collect the voltage result V.

   - Process the time dependent devices control. i.e Batteries state of charge.

4. End.
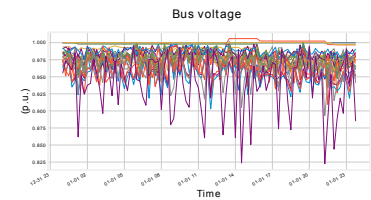
Figure 9.1: Example of voltage results for a time series simulation.

10 Stochastic power flow

The stochastic power flow is a simulation mode that runs power flows for several randomly generated states of the grid. Those states are defined by combinations of load and generation random samples and a fix connectivity state. The goal is to obtain the probability distribution of the voltages and the branch loading given the probability distribution of the load and the generation.

$$CDF_{voltage} = f(CDF_{power})$$

Mathematically speaking, we want to obtain the cumulative frequency distribution (CDF) of the voltages, as a function of the CDF of the power injections. The transformation function chosen is the power flow.

Varying the connectivity is definitely possible but it will interfere with the purpose of this simulation. If there is the need to vary the connectivity, then it is recommended to run a stochastic simulation for each connectivity state.

10.1 Cumulative Distribution Function (CDF)

Usually, one finds that tools model loads and generation with probability distribution functions (PDF) such as the normal or Weibull distributions. In the author's opinion there is a much easier and correct way: To use only data-based cumulative distribution functions (CDF). This allows you to build sampling artefacts from any data source real or theoretical.

Building a CDF   Suppose that you have recorded load data from a smart meter or any other device. Suppose that there are a thousand values in the record.

To build a CDF:

- Sort the records by value, this is the y value from the CDF. $data = sort(records)$

- The associated probabilities are an array of equal length. These are given by the expression:

$$p_i = \frac{i}{n-1} \quad \forall n > 1, i \in 0..n-1$$

Simple example   If the records are:

$$records = [0.2, 0.5, 0.8, 0.3, 0.1, 0.2, 0.3, 0.5, 0.7]$$

The CDF is:

$$CDF_{data} = [0.1, 0.2, 0.2, 0.3, 0.3, 0.5, 0.5, 0.7, 0.8]$$

$$CDF_{probabilities} = [0, 0.125, 0.25, 0.375, 0.5, 0.625, 0.875, 1.0]$$

Sampling the CDF   Once constructed, we can draw infinite samples from the CDF in a very simple way.

First we need to have an interpolation function (linear, quadratic, cubic, spline, ..., at your choice)

Then, We need a uniformly distributed random number generator ($rand$ usually)

With these, we get a new load value from the CDF, by interpolating with a randomly generated probability.

$$p_{new} = rand(0,1)$$

$$value_{new} = interpolate(CDF_{probabilities}, CDF_{data}, p_{new})$$

10.2   Monte Carlo

The Monte Carlo simulation was invented by Stanislaw Ulam and John von Neumann to simulate the diffusion of uranium electrons while developing the nuclear bomb. This very simple algorithm has proven to be an incredibly powerful tool to estimate uncertain behaviours, and it is the de-facto algorithm to simulate stochastic power flows.

The underlying idea is to draw n samples and compute the average. When the average's standard deviation becomes sufficiently small, it indicates that enough samples have been drawn to characterize the system.

In our case, we will draw load and generation samples, evaluate the voltage by running a power flow with those samples and calculate the voltage mean and standard deviation. When the standard deviation becomes small enough (a tolerance value that we set) we stop sampling. See the figure 10.3.

Algorithm

1. Construct the CDF of all the loads and generators from their profiles of data. For the loads we construct a DSF for the active power and another for the reactive power, for the generators, we only need the CDF for the active power.

2. Initialize control variables: $iterations = 0$, $V_{summation} = 0$

3. While the variance is less than a tolerance value:

    (a) Construct the injection vectors S and I, sampling from the CDF.

    (b) Run a power flow (see chapter 7) with those arrays and get the voltages vector V.

    (c) Increase the number of iterations: $iterations += 1$

    (d) Compute the voltages summation: $V_{summation} += V$
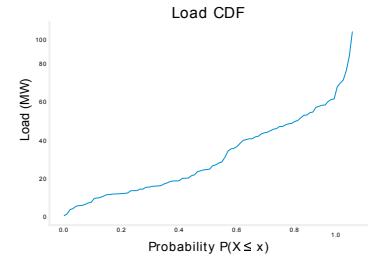
Figure 10.1: Example of cumulative distribution function of a load.
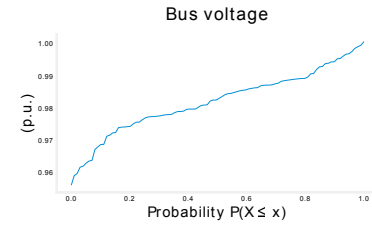
Figure 10.2: Example of cumulative distribution function of the voltage. This is the Result of a probabilistic power flow.
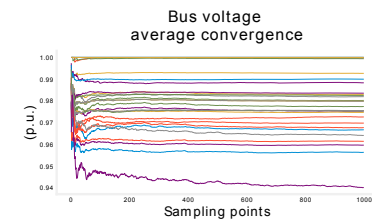
Figure 10.3: Example of voltage convergence. This is the Result of a probabilistic power flow.

(e) Compute the voltage average: $V_{mean} = V_{summation}/iterations$

(f) Compute the variance: $min((V - V_{mean})^2/(iterations - 1))$

(g) Store all the values that you want to have at the end of the loop.

4. End.

11 *WLS State estimation*

Weighted Least Squares State estimation is a method to correct field measurements with the mathematical grid equilibrium equations (See equation 7.1). In this chapter we are going to follow the implementation suggested in the book from Antonio Gómez Expósito and Ali Abur [1].

The estate estimation method is formulated with the following recursive linear system:

$$\left[H^T \times W \times H + \delta \cdot diag(H^T \times W \times H) \right] \times \Delta x = \left[H^T \times W \times (z - h) \right]$$
(11.1)

Where:

- H: Jacobian matrix for the estate estimation problem.

- W: Weights diagonal matrix.

- δ: Step control value. It is non negative.

- x: vector of solutions (voltage modules and angles).

- z: field measurements vector.

- h: calculated magnitudes matching the measurements structure.

The linear system presented in equation 11.1 is solved using the Levenberg-Marquardt method (See 7.3.4) and alternatively using the Newton-Raphson method, but the later is less suitable.

Weights matrix W   The weights matrix is a diagonal matrix of size equal to the number of measurements, where the diagonal values are the inverse of the variance (σ^2) of the measurements.

$$W_{i,i} = \frac{1}{\sigma_i^2} \quad \forall i \in Measurements$$
(11.2)

Jacobian matrix H   The Jacobian matrix for the state estimation method is found as follows:

$$H = \begin{bmatrix} A1 & A2 \\ B1 & B2 \\ C1 & C2 \\ D1 & D2 \\ E1 & E2 \\ F1 & F2 \end{bmatrix} \tag{11.3}$$

Note that the Jacobian formulated here is composed of 12 real-valued submatrices. As you might have noticed we try to avoid summation based formulations as much as possible, so to compute the submatrices we will need to compute the following complex derivatives. Let's introduce them first:

The derivatives computation using matrix formulation has been taken from .
  Ray D Zimmerman. Ac power flows, generalized opf costs and their derivatives using complex matrix notation, 2010

$$\left[\frac{\partial S_{inj}}{\partial \theta}\right] = j \cdot [V_{diag}] \times ([I_{bus,diag}] - [Y_{bus}] \times [V_{diag}])^* \tag{11.4}$$

$$\left[\frac{\partial S_{inj}}{\partial Vm}\right] = j \cdot [E_{diag}] \times ([I_{bus,diag}] + [Y_{bus}] \times [V_{diag}])^* \tag{11.5}$$

All these derivatives are square matrices in complex numbers.

$$\left[\frac{\partial S_{flow}}{\partial \theta}\right] = j \cdot ([I_{f,diag}]^* \times [C_f] \times [V_{diag}] - diag([C_f] \times [V]) \times [Y_f]^* \times [V_{diag}]^*) \tag{11.6}$$

$$\left[\frac{\partial S_{flow}}{\partial Vm}\right] = [I_{f,diag}]^* \times [C_f] \times [E_{diag}] - diag([C_f] \times [V]) \times [Y_f]^* \times [E_{diag}]^* \tag{11.7}$$

The matrices with the subindex $diag$ are diagonal matrices constructed from the vector with the same name. For instance $[V_{diag}]$ is a diagonal matrix formed with the voltages vector $[V]$.

$$\left[\frac{\partial I_{mag}}{\partial \theta}\right] = j \cdot [Y_f] \times [V_{diag}] \tag{11.8}$$

$$\left[\frac{\partial I_{mag}}{\partial Vm}\right] = j \cdot [Y_f] \times [E_{diag}] \tag{11.9}$$

The matrices used for the derivatives computation and the calculated magnitudes vector (h) are:

$$[E] = \frac{[V]}{[Vm]} \tag{11.10}$$

$$[V_f] = [C_f] \times [V] \tag{11.11}$$

$$[Y_f] = [C_f] \times [Y_{bus}] \tag{11.12}$$

$$[I_f] = [Y_f] \times [V] \tag{11.13}$$

$$[I_{inj}] = [Y_{bus}] \times [V] + [I_{bus}] \tag{11.14}$$

$$[S_{inj}] = [V] \cdot [I_{inj}]^* \tag{11.15}$$

$$[S_f] = [V_f] \cdot [I_f]^* \tag{11.16}$$

The submatrices $[A1]$, $[A2]$, $[B1]...[F2]$, to be used in the Jacobian $[H]$ are not the ones with all the entries, instead we have to use sub-matrices of the real or imaginary parts of the previously introduced derivative matrices where the rows to keep are the ones with the indices of the nodes or branches with measurements of the respective magnitudes and the columns to keep are the ones with the indices of the non slack nodes for $[A1]$, $[B1]$, $[C1]$, $[D1]$, $[E1]$ and $[F1]$ and all the columns for the others. Therefore the submatrices are:

$$[A1] = Re\left\{ \left[\frac{\partial S_{flow}}{\partial Vm} \right] \right\}_{[np_{flow},pqpv]} \tag{11.17}$$

np_{flow} is a vector of branch indices where there are active power flow measurements.

$$[A2] = Re\left\{ \left[\frac{\partial S_{flow}}{\partial \theta} \right] \right\}_{[np_{flow},:]} \tag{11.18}$$

$$[B1] = Re\left\{ \left[\frac{\partial S_{inj}}{\partial Vm} \right] \right\}_{[np_{inj},pqpv]} \tag{11.19}$$

np_{inj} is a vector of node indices where there are active power injection measurements.

$$[B2] = Re\left\{ \left[\frac{\partial S_{inj}}{\partial \theta} \right] \right\}_{[np_{inj},:]} \tag{11.20}$$

$$[C1] = Im\left\{ \left[\frac{\partial S_{flow}}{\partial Vm} \right] \right\}_{[nq_{flow},pqpv]} \tag{11.21}$$

nq_{flow} is a vector of branch indices where there are reactive power flow measurements.

$$[C2] = Im\left\{ \left[\frac{\partial S_{flow}}{\partial \theta} \right] \right\}_{[nq_{flow},:]} \tag{11.22}$$

$$[D1] = Im\left\{ \left[\frac{\partial S_{inj}}{\partial Vm} \right] \right\}_{[nq_{inj},pqpv]} \tag{11.23}$$

nq_{inj} is a vector of node indices where there are reactive power injection measurements.

$$[D2] = Im \left\{ \left[\frac{\partial S_{inj}}{\partial \theta} \right] \right\}_{[nq_{inj},:]} \qquad (11.24)$$

$$[E1] = abs \left(\left[\frac{\partial I_{flow}}{\partial Vm} \right] \right)_{[nI_{flow}, pqpv]} \qquad (11.25)$$

nI_{flow} is a vector of branch indices where there are current flow module measurements.

$$[E2] = abs \left(\left[\frac{\partial I_{flow}}{\partial \theta} \right] \right)_{[nI_{flow},:]} \qquad (11.26)$$

$$[F1] = \left[\frac{\partial Vm}{\partial Vm} \right]_{[nv, pqpv]} = diag([1])_{[nv, pqpv]} \qquad (11.27)$$

nv is a vector of node indices where there are voltage module measurements.

$$[F2] = \left[\frac{\partial Vm}{\partial \theta} \right]_{[nv,:]} = [0]_{[nv,:]} \qquad (11.28)$$

Calculated measurements h   To compute the residual ($z - h$) we need to compute the vector h. As the Jacobian, this vector is composed of 6 subvectors matching the Jacobian structure.

$$[h] = \begin{bmatrix} a3 \\ b3 \\ c3 \\ d3 \\ e3 \\ f3 \end{bmatrix} \qquad (11.29)$$

$$[a3] = Re \left\{ [S_f] \right\}_{[np_{flow}]} \qquad (11.30)$$

np_{flow} is a vector of branch indices where there are active power flow measurements.

$$[b3] = Re \left\{ [S_{inj}] \right\}_{[np_{inj}]} \qquad (11.31)$$

np_{inj} is a vector of node indices where there are active power injection measurements.

$$[c3] = Im \left\{ [S_f] \right\}_{[nq_{flow}]} \qquad (11.32)$$

nq_{flow} is a vector of branch indices where there are reactive power flow measurements.

$$[d3] = Im \left\{ [S_{inj}] \right\}_{[nq_{inj}]} \qquad (11.33)$$

nq_{inj} is a vector of node indices where there are reactive power injection measurements.

$$[e3] = abs \left([I_f] \right)_{[nI_{flow}]} \qquad (11.34)$$

nI_{flow} is a vector of branch indices where there are current flow module measurements.

$$[f3] = [Vm]_{[nv]} \qquad (11.35)$$

nv is a vector of node indices where there are voltage module measurements. Note that Vm is the module of the last iteration voltage solution.

The solution vector Δx The linear system solution will produce a vector $[\Delta x]$ at each iteration. This vector contains the increments of the voltage angle ($\Delta\theta$) extended with the increments of the voltage modules ($\Delta V m$) for all the nodes including the slack.

$$[\Delta x] = \begin{bmatrix} \Delta\theta_{(pqpv)} \\ \Delta Vm \end{bmatrix} \tag{11.36}$$

In this fashion we can split the increments vector:

$$[\Delta\theta]_{pqpv} = [\Delta x]_{[:npqpv]} \tag{11.37}$$

, from there until the end we find the voltage module increments.

The angle increments go from the first until the index equal to the number of pq and pv nodes $npqpv$

$$[\Delta Vm] = [\Delta x]_{[npqpv::]} \tag{11.38}$$

To compose the voltage solution at the iteration $k+1$:

$$[V]^{(k+1)} = [V]^{(k} + [\Delta Vm]^{-j\cdot[\Delta\theta]} \tag{11.39}$$

The outcome of the method are the node voltages vector $[V]$. From this voltage solution, one shall compose the branch power flows using the equations described in 7.7.

11.0.1   *Augmented Hatchel matrix method*

The augmented Hatchel matrix method, introduces two advantages:

- It allows virtual measurements with zero uncertainty (values coming from a power flow solution for instance)

- The linear system has a better condition number, and therefore has better convergence properties.

The linear system to solve iteratively in te Hatchel method is:

$$\begin{bmatrix} \frac{R}{\alpha} & H & 0 \\ H^T & 0 & C^T \\ 0 & C & 0 \end{bmatrix} \times \begin{bmatrix} \mu \\ \Delta x \\ \lambda \end{bmatrix} = \begin{bmatrix} z - h \\ 0 \\ -c(x) \end{bmatrix} \tag{11.40}$$

Where:

- H: Jacobian matrix for the estate estimation problem of the measurements with uncertainty. (See 11.3)

- C: Jacobian matrix for the estate estimation problem of the measurements without uncertainty. (See 11.3)

- R: Matrix of standard deviations. It is the inverse of the weights diagonal matrix.(See 11.2)

- α: Value to condition the matrix.

- x: vector of solutions.

- λ: Lagrange multipliers.

- μ: Lagrange multipliers.

- z: field measurements vector.

- h: calculated magnitudes matching the measurements structure for the measurements with uncertainty. (See 11.29)

- $c(x)$: calculated magnitudes matching the measurements structure for the measurements without uncertainty.

Conditioning value (α)

$$\alpha = \frac{1}{max(W_{i,i})} \tag{11.41}$$

Matrix of standard deviations (R)

$$R_{i,i} = \sigma_{i,i}^2 \tag{11.42}$$

A Example grid

After having introduced the basics, this appendix defines a simple grid to help fine tuning the programmed algorithms. The grid used will be the 5 Bus grid presented in [1].

[1] Lynn Powell. *Power system load flow analysis*. McGraw Hill professional, 2004

Figure A.1: 5 bus grid with terminals and switches.
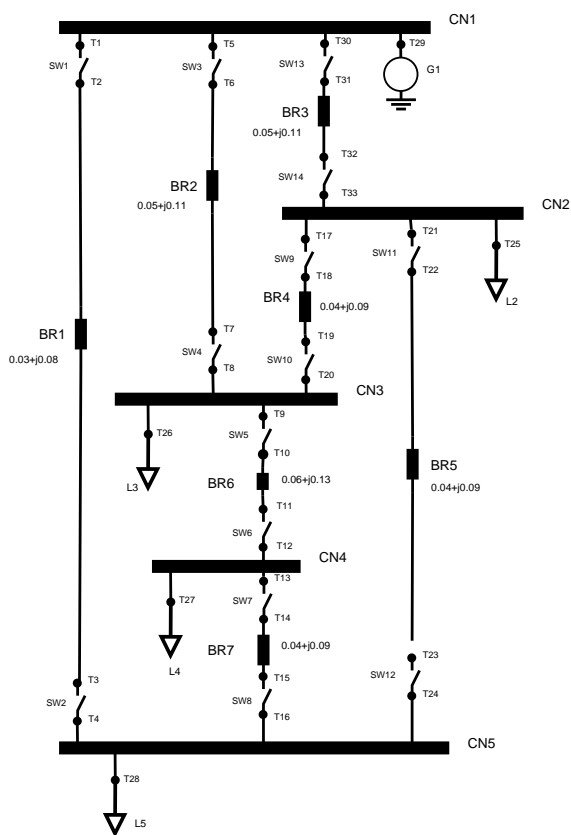
The grid is defined with the following data:

Name	is slack	Vnom	Vmin	Vmax
CN1	False	230.0	0.9	1.1
CN2	False	230.0	0.9	1.1
CN3	False	230.0	0.9	1.1
CN4	True	230.0	0.9	1.1
CN5	False	230.0	0.9	1.1

Table A.1: Bus data

Name	bus from	bus to	rate	R	X	G	B
Branch 0-1	Bus 0	Bus 1	400	0.00281	0.0281	0	0.00712
Branch 0-3	Bus 0	Bus 3	224.4	0.00304	0.0304	0	0.00658
Branch 0-4	Bus 0	Bus 4	273.3	0.00064	0.0064	0	0.03126
Branch 1-2	Bus 1	Bus 2	128.3	0.00108	0.0108	0	0.01852
Branch 2-3	Bus 2	Bus 3	34.6	0.00297	0.0297	0	0.00674
Branch 3-4	Bus 3	Bus 4	240	0.00297	0.0297	0	0.00674

Table A.2: Branch data

Name	Terminal	Z	I	S
L2	T25	0j	0j	40 + 20j
L3	T26	0j	0j	25 + 15j
L4	T27	0j	0j	40 + 20j
L5	T28	0j	0j	50 + 20j

Table A.3: Load data

A.1   Grid compilation results

First the topological compilation results:

———————————————————- RESULTS ———————————————————-

The calculated node results in positive sequence are:

$$S_{bus,1} = \begin{bmatrix} 1.55 + 0j & -0.4 - 0.2j & -0.25 - 0.15j & -0.4 - 0.2j & -0.5 - 0.2j \end{bmatrix} \, p.u.$$

$$V_{bus,1} = \begin{bmatrix} 1 + 0j, & 1 + 0j, & 1 + 0j, & 1 + 0j, & 1 + 0j \end{bmatrix} \, p.u.$$

$$Types = \begin{bmatrix} Slack, & PQ, & PQ, & PQ, & PQ \end{bmatrix} \, p.u.$$

$$Y_{bus,1} = \begin{bmatrix} 10.959 - 25.997j & -3.425 + 7.534j & -3.425 + 7.534j & 0. + 0.j & -4.11 + 10.959j \\ -3.425 + 7.534j & 11.672 - 26.061j & -4.124 + 9.278j & 0. + 0.j & -4.124 + 9.278j \\ -3.425 + 7.534j & -4.124 + 9.278j & 10.475 - 23.119j & -2.927 + 6.341j & 0. + 0.j \\ 0. + 0.j & 0. + 0.j & -2.927 + 6.341j & 7.051 - 15.595j & -4.124 + 9.278j \\ -4.11 + 10.959j & -4.124 + 9.278j & 0. + 0.j & -4.124 + 9.278j & 12.357 - 29.486j \end{bmatrix} \, p.u.$$

name	bus	P	Vset	Snom	Qmin	Qmax
Gen 0a	Bus 0	40	1	9999	-30	30
Gen 0b	Bus 0	170	1	9999	0	127.5
Gen 2	Bus 2	323.49	1	9999	-390	390
Gen 3	Bus 3	0	1	9999	0	150
Gen 4	Bus 4	466.51	1	9999	-450	450

Table A.4: Generator data

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20	T21	T22	T23	T24	T25	T26	T27	T28	T29	T30	T31	T32	T33
CN1	1				1																								1	1			1
CN2																	1				1				1								1
CN3						1	1											1							1								
CN4			1									1	1				1							1			1						
CN5				1													1							1			1						

Table A.5: $C_{bus,terminal}$

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20	T21	T22	T23	T24	T25	T26	T27	T28	T29	T30	T31	T32	T33
BR1		1																															
BR2						1																											
BR3																														1			
BR4																1																	
BR5																				1													
BR6										1																							
BR7														1																			

Table A.6: $C_{branch\_f,terminal}$

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20	T21	T22	T23	T24	T25	T26	T27	T28	T29	T30	T31	T32	T33
BR1				1																													
BR2								1																									
BR3																														1			
BR4																		1															
BR5																				1													
BR6											1																						
BR7														1																			

Table A.7: $C_{branch\_t,terminal}$

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20	T21	T22	T23	T24	T25	T26	T27	T28	T29	T30	T31	T32	T33
SW1	1	1																															
SW2		1	1																														
SW3				1	1																												
SW4						1	1																										
SW5								1	1																								
SW6										1	1																						
SW7												1	1																				
SW8															1	1																	
SW9																	1	1															
SW10																			1	1													
SW11																					1	1											
SW12																							1	1									
SW13																													1	1			
SW14																															1	1	

Table A.8: $[C_{switch,terminal}]$

	SW1	SW2	SW3	SW4	SW5	SW6	SW7	SW8	SW9	SW10	SW11	SW12	SW13	SW14
States	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table A.9: $[switch\_states]$

	CN1	CN2	CN3	CN4	CN5
L2		1			
L3			1		
L4				1	
L5					1

Table A.10: $[C_{load,bus}]$

	CN1	CN2	CN3	CN4	CN5
G1	1				

Table A.11: $[C_{generation,bus}]$

	CN1	CN2	CN3	CN4	CN5
BR1	1				1
BR2	1		1		
BR3	1	1			
BR4		1	1		
BR5		1			1
BR6			1	1	
BR7				1	1

Table A.12: $[C]$

	CN1	CN2	CN3	CN4	CN5
CN1	1	1	1		1
CN2	1	1	1		1
CN3	1	1	1	1	
CN4			1	1	1
CN5	1	1		1	1

Table A.13: $[C_{bus,bus}]$

Bibliography

Compressed column storage (ccs). URL `http://netlib.org/linalg/html_templates/node92.html`.

Jos Arrillaga and CP Arnold. *Computer analysis of power systems*. Wiley Online Library, 1990.

Florian Dorfler and Francesco Bullo. Kron reduction of graphs with applications to electrical networks. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 60(1):150–163, 2013.

Charles L Fortescue. Method of symmetrical co-ordinates applied to the solution of polyphase networks. *Transactions of the American Institute of Electrical Engineers*, 37(2):1027–1140, 1918.

Antonio Gomez-Exposito and Ali Abur. *Power system state estimation: theory and implementation*. CRC press, 2004.

S Iwamoto and Y Tamura. A load flow calculation method for ill-conditioned power systems. *IEEE transactions on power apparatus and systems*, (4): 1736–1743, 1981.

William H Kersting. *Distribution system modeling and analysis*. CRC press, 2012.

Chengxi Liu, Bin Wang, Fengkai Hu, Kai Sun, and Claus Leth Bak. Online voltage stability assessment for load areas based on the holomorphic embedding method. *IEEE Transactions on Power Systems*, 2017.

Lynn Powell. *Power system load flow analysis*. McGraw Hill professional, 2004.

Priscila Rossoni, William Moreti da Rosa, and Edmarcio Antonio Belati. Linearized ac load flow applied to analysis in electric power systems. *IEEE Latin America Transactions*, 14(9):4048–4053, 2016.

Muthu Kumar Subramanian. Application of holomorphic embedding to the power-flow problem, 2014.

Joshua Adam Taylor. *Convex optimization of power systems*. Cambridge University Press, 2015.

A. Trias. The holomorphic embedding load flow method. pages 1–8, July 2012. ISSN 1944-9925. DOI: 10.1109/PESGM.2012.6344759.

Jinquan Zhao, Hsiao-Dong Chiang, Ping Ju, and Hua Li. On pv-pq bus type switching logic in power flow computation. In *Power Systems Computation Conference (PSCC), Glasgow, Scotland*, 2008.

Ray D Zimmerman. Ac power flows, generalized opf costs and their derivatives using complex matrix notation, 2010.