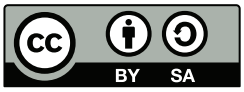
Santiago Peñate Vera

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 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.
- $A^{T} \rightarrow$ 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.
- $[1]$: Matrix of ones.
- $[0]$: Matrix of zeros.

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

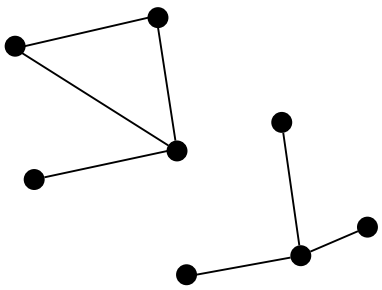


Figure 3.1: Graph with 8 nodes and 7 branches. The graph contains two islands.

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.

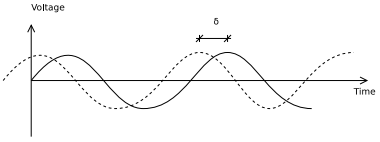


Figure 4.1: Voltage delay.

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 δ 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.

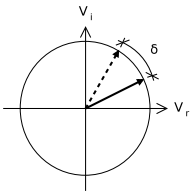


Figure 4.2: Voltage delay in phasor representation in the complex plane.

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)		
	δ (Voltage angle)		$V = V_m \cdot e^{\delta}$
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.

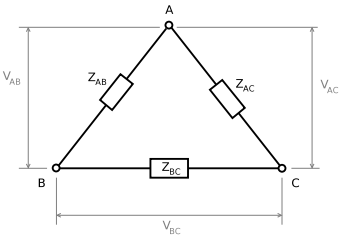


Figure 4.3: Delta connection scheme.

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.

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

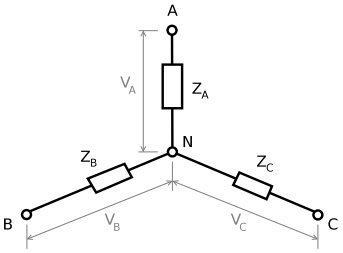


Figure 4.4: Wye connection scheme.

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

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 $S_{Base} = 100MVA$.

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)	$Z_{Base} = V^2_{Base}/S_{Base}$
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*.

[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

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:

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

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

Where a is the transformation eigenvector:

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

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

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:

$$Z_{seq} = A_s^{-1} \times Z_{ABC} \times A_s \tag{4.11}$$

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:

$$\begin{aligned} Z_0 &= 0.7735 + j1.9373 \\ Z_1 &= 0.3061 + j0.6270 \\ Z_2 &= 0.3061 + j0.6270 \end{aligned}$$

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:

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

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

$$Z_{ABC_{approx}} = \begin{bmatrix} Z_0 + Z_1 + Z_2 & Z_0 + aZ_1 + a^2Z_2 & Z_0 + a^2Z_1 + aZ_2 \\ Z_0 + a^2Z_1 + aZ_2 & Z_0 + Z_1 + Z_2 & Z_0 + aZ_1 + a^2Z_2 \\ Z_0 + aZ_1 + a^2Z_2 & Z_0 + a^2Z_1 + aZ_2 & Z_0 + Z_1 + Z_2 \end{bmatrix} \tag{4.13}$$

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 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.

5.1 Π Model

The pi model is composed by a series admittance $\mathbf{Y}_{serie}$ and a shunt admittance $\mathbf{Y}_{sh}$ 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 $\mathbf{Y}_{serie}$ is computed as the inverse of $\mathbf{Z}_{ABC}$. From the line's calculation it is obtained the series impedance matrix $\mathbf{Z}_{ABC}$ and the shunt admittance matrix $\mathbf{Y}_{sh}$.

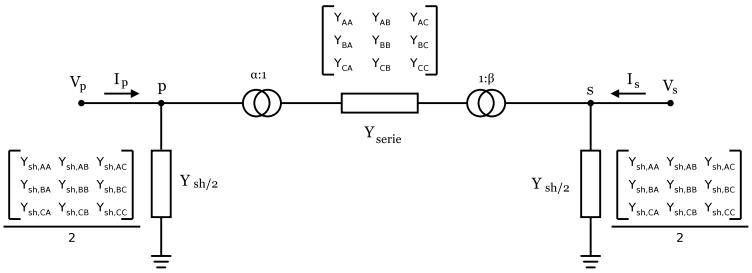


Figure 5.1: General branch model.

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

$$\begin{bmatrix} \mathbf{I}_p \\ \mathbf{I}_s \end{bmatrix} = \begin{bmatrix} \mathbf{Y}_{pp} & \mathbf{Y}_{ps} \\ \mathbf{Y}_{sp} & \mathbf{Y}_{ss} \end{bmatrix} \times \begin{bmatrix} \mathbf{V}_p \\ \mathbf{V}_s \end{bmatrix} \tag{5.1}$$

Where:

$\mathbf{Y}_{pp}$	$\mathbf{Y}_{ps}$	$\mathbf{Y}_{sp}$	$\mathbf{Y}_{ss}$
$\frac{\mathbf{Y}_{series}+\frac{\mathbf{Y}_{sh}}{2}}{\alpha^2}$	$\frac{-\mathbf{Y}_{series}}{\alpha\beta}$	$\frac{-\mathbf{Y}_{series}}{\alpha\beta}$	$\frac{\mathbf{Y}_{series}+\frac{\mathbf{Y}_{sh}}{2}}{\beta^2}$

Table 5.1: Equations of the generalized Π model.

The formula 5.1 will be used to compute the network admittance matrices.

5.1.1 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 admit-

tance matrix of the branch is similar to the one presented for three phases, but each sub-matrix has only one element (one equivalent phase).

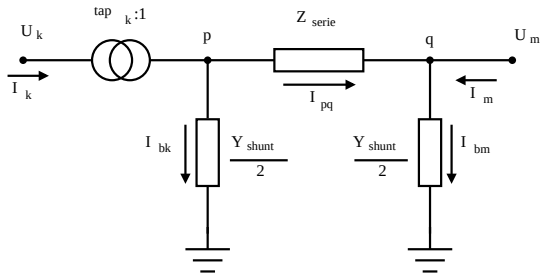


Figure 5.2: General positive sequence branch model.

- 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.

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.

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

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

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.

$$Y_{branch} = \begin{bmatrix} Y_{ff} & Y_{ft} \\ Y_{tf} & Y_{tt} \end{bmatrix}$$

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 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}.

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

$$\begin{bmatrix} 0.0112 - j0.0232 & -0.0054 + j0.008 & -0.0020 + j0.0052 \\ -0.0054 + j0.008 & 0.0125 - j0.024 & -0.0033 + j0.0063 \\ -0.0020 + j0.0052 & -0.0033 + j0.0063 & 0.0100 - j0.0224 \end{bmatrix} \quad p.u. \tag{5.2}$$

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

$$\begin{bmatrix} -0.0112 + j0.0232 & 0.0054 - j0.008 & 0.0020 - j0.0052 \\ 0.0054 - j0.008 & -0.0125 + j0.024 & 0.0033 - j0.0063 \\ 0.0020 - j0.0052 & 0.0033 - j0.0063 & -0.0100 + j0.0224 \end{bmatrix} \quad p.u. \tag{5.3}$$

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

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

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

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

5.3 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 5.2: Three-phase transformer impedances of the branch model.

The table 5.2 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

$$Y_I = \begin{bmatrix} y_t & 0 & 0 \\ 0 & y_t & 0 \\ 0 & 0 & y_t \end{bmatrix} \tag{5.4}$$

$$Y_{II} = \begin{bmatrix} 2y_t & -y_t & -y_t \\ -y_t & 2y_t & -y_t \\ -y_t & -y_t & 2y_t \end{bmatrix} \tag{5.5}$$

$$Y_{III} = \begin{bmatrix} -y_t & y_t & 0 \\ 0 & -y_t & y_t \\ y_t & 0 & -y_t \end{bmatrix} \tag{5.6}$$

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:

$$y_t = \frac{3}{r_l + jx_l} \tag{5.7}$$

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

5.3.1 *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):

$$Zn_{hv} = U_{hv}^2 / S_n \tag{5.8}$$

Nominal impedance LV (Ohm):

$$Zn_{lv} = U_{lv}^2 / S_n \tag{5.9}$$

Short circuit impedance (p.u.):

$$z_{sc} = U_{sc}/100 \tag{5.10}$$

Short circuit resistance (p.u.):

$$r_{sc} = \frac{P_{cu}/1000}{S_n} \tag{5.11}$$

Short circuit reactance (p.u.):

$$x_{sc} = \sqrt{z_{sc}^2 - r_{sc}^2} \tag{5.12}$$

HV resistance (p.u.):

$$r_{cu,hv} = r_{sc} \cdot GR_{hv1} \tag{5.13}$$

LV resistance (p.u.):

$$r_{cu,lv} = r_{sc} \cdot (1 - GR_{hv1}) \tag{5.14}$$

HV shunt reactance (p.u.):

$$xs_{hv} = x_{sc} \cdot GX_{hv1} \tag{5.15}$$

LV shunt reactance (p.u.):

$$xs_{lv} = x_{sc} \cdot (1 - GX_{hv1}) \tag{5.16}$$

Shunt resistance (p.u.):

$$r_{fe} = \frac{Sn}{P_{fe}/1000} \tag{5.17}$$

Magnetization impedance (p.u.):

$$z_m = \frac{1}{I_0/100} \tag{5.18}$$

Magnetization reactance (p.u.):

$$x_m = \frac{1}{\sqrt{\frac{1}{z_m^2} - \frac{1}{r_{fe}^2}}} \tag{5.19}$$

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):

$$z_{series} = Z_m = r_{sc} + j \cdot x_{sc} \tag{5.20}$$

The series admittance is [p.u.]:

$$y_{series} = \frac{1}{z_{series}} \tag{5.21}$$

Leakage impedance (or shunt impedance):

$$Z_l = r_{fe} + j \cdot x_m \tag{5.22}$$

Shunt admittance [p.u.]:

$$y_{shunt} = 1/Z_l \tag{5.23}$$

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

$$y_t = \frac{3}{z_{series}} \tag{5.24}$$

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

Example Let's consider a distribution transformer with the following nameplate characteristics:

- Primary connection: Δ
- Secondary connection: Yg
- $S_n = 0.5MVA$
- $U_{hv} = 20kV$
- $U_{lv} = 0.4kV$
- $U_{sc} = 6\%$
- $p_{cu} = 6kW$
- $p_{fe} = 1.4kW$
- $I_0 = 0.28\%$
- $GR_{hv} = 0.5$
- $GX_{hv} = 0.5$

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 5.2:

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

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

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

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

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.

5.4 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 Example grid

After having introduced the basics, this chapter defines a simple grid for which all the calculations are illustrated. The grid used will be the IEEE 5 Bus standard grid, which is defined with the following data:

name	active	is slack	Vnom	Vmin	Vmax
Bus 0	True	False	230.0	0.9	1.1
Bus 1	True	False	230.0	0.9	1.1
Bus 2	True	False	230.0	0.9	1.1
Bus 3	True	True	230.0	0.9	1.1
Bus 4	True	False	230.0	0.9	1.1

Table 6.1: Bus data

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

Table 6.2: Branch data

name	bus	active	Z	I	S
Load 1	Bus 1	True	0j	0j	300 + 98.61j
Load 2	Bus 2	True	0j	0j	300 + 98.61j
Load 3	Bus 3	True	0j	0j	99.99 + 99.99j

Table 6.3: Load data

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

Table 6.4: Generator data

This data will be used in the next chapters to compute examples of the magnitudes.

6.1 Grid compilation results

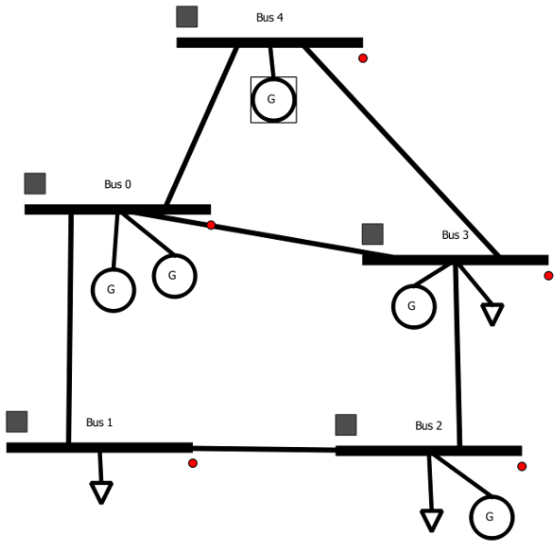


Figure 6.1: IEEE 5 bus standard grid. Representation generated with GridCal.

The calculated node results are:

Bus	Power (p.u.)	Voltage (p.u.)	Current (p.u.)	Bus types
Bus 0	2.1+0j	1+0j	0j	2
Bus 1	-3-0.9861j	1+0j	0j	1
Bus 2	0.2349-0.9861j	1+0j	0j	2
Bus 3	-0.9999-0.9999j	1+0j	0j	3
Bus 4	4.6651+0j	1+0j	0j	2

Table 6.5: Node results

	Bus 0	Bus 1	Bus 2	Bus 3	Bus 4
Bus 0	22.25-222.48j	-3.525+35.23j	0j	-3.26+32.57j	-15.47+154.70j
Bus 1	-3.52+35.23j	12.69-126.9j	-9.17+91.68j	0j	0j
Bus 2	0j	-9.17+91.68j	12.50-125.0j	-3.33+33.34j	0j
Bus 3	-3.26+32.57j	0j	-3.33+33.34j	9.92-99.23j	-3.33+33.34j
Bus 4	-15.47+154.70j)	0j	0j	-3.34+33.34j	18.80-188.02j

Table 6.6: Admittance matrix

7 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.)

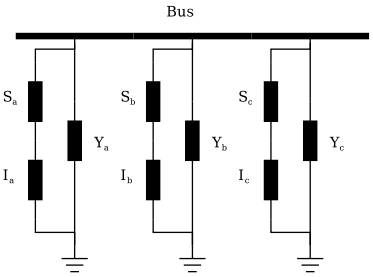


Figure 7.1: $YISV$ Bus model.

7.1 The substation

In reality a substation is a collection of real bus bars (metal bars usually made of copper or aluminium) switchgear and shunt devices. The use of a substation is to change the connectivity topology of the grid.

For calculation, the substation connectivity is solved first, originating one or more calculation buses. Therefore, the substations are irrelevant at calculation level, but they are very relevant in the definition of the grid topology prior to the calculation.

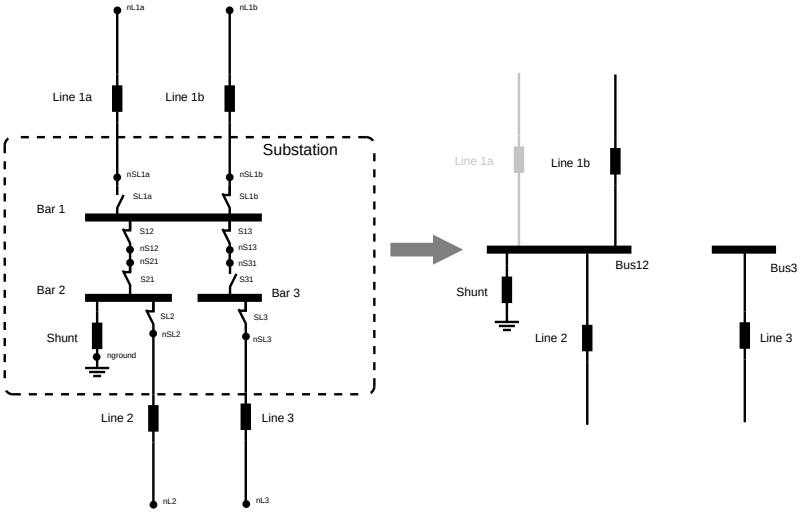


Figure 7.2: Substation example. The substation becomes a calculation bus.

In the example depicted in the figure 7.2 it is shown how a node system is reduced from three real buses to two calculation buses by eliminating the non connected elements and associating the connected ones. The buses1, 2 are connected because the switches between them are closed, hence the three buses become one. the bus 3 is isolated from the others, but still connected to a line. The lines 1b, 2 and 3 are also connected through

closed switches, hence those remain connected to their respective buses. However the line 1a is connected through an open switch, therefore is left out for calculation purposes.

In order to calculate this reduction, we need to introduce extra nodes in the substation model. These are topological nodes that will help us to perform the connectivity calculation. Then we represent the substation as a node-branch graph:

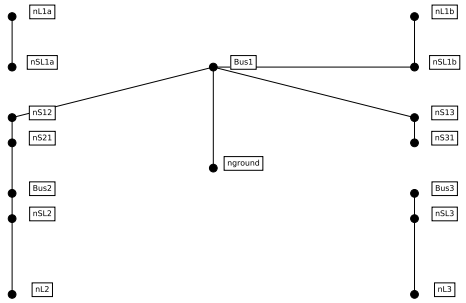


Figure 7.3: Substation example graph.

Observe the islands formed:

- Island 1: **Bus1**, **Bus2**, nL1b, nL2, nS12, nS13, nS21, nS31, nSL1b, nSL2, nground.
- Island 2: **Bus3**, nL3, nSL3.
- Island 3: nL1a, nSL1a.

Removing the topological nodes, only two islands remain:

- Island 1: **Bus1, Bus2**. -> these two form a joint calculation node.
- Island 2: **Bus3**.

Now we only need to check which non topological branches are in every island and associate them to the formed calculation node.

7.2 *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).

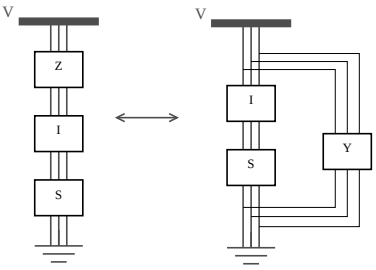


Figure 7.4: ZIP / $YISV$ device model.

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

Table 7.1: Main properties of the static device object. More properties my be added if needed.

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 8.3.

7.3 *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

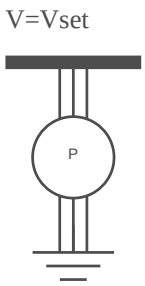


Figure 7.5: PV device model.

8 Topology analysis and consolidation

8.1 Islands detection

The admittance matrix of a circuit with islands is singular. Therefore, the circuit it cannot be solved. To overcome this issue, we have to divide the circuit into fully connected islands for calculation. The algorithm to achieve this is the Breadth First Search (BFS) algorithm.

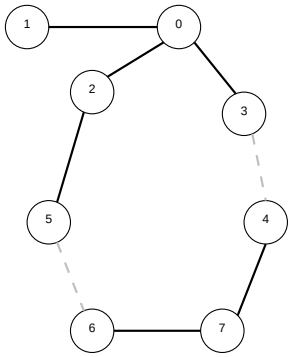


Figure 8.1: Graph with two islands.

A c++ implementation of a Graph with functions to detect the islands (`connectedComponents`) is as follows:

```
class Graph {

/**
* @brief Number of vertices
*/
std::size_t V;

/**
* @brief Array of lists containing adjacency (this is the graph)
*/
std::list<std::size_t> *adj;

/**
* @brief A function used by DFS
* @param v index of the node
* @param visited array of visited nodes
* @param islands structure to map the islands
* @param island_idx index of the island
*/
```

```
void DFSUtil(std::size_t v, bool visited[],
 std::vector<std::vector<std::size_t>> *islands,
 std::size_t island_idx) {

// Mark the current node as visited and prlong it
visited[v] = true;

// add element to the island
islands->at(island_idx).push_back(v);

// Recur for all the vertices adjacent to this vertex
std::list<std::size_t>::iterator i;
for(i = adj[v].begin(); i != adj[v].end(); ++i)
if(!visited[*i])
DFSUtil(*i, visited, islands, island_idx);
}

public:

/**
* @brief Graph constructor
* @param V number of vertices
*/
Graph(std::size_t vertex_number){
V = vertex_number;
adj = new std::list<std::size_t>[vertex_number];
}

/**
* @brief Graph destructor
*/
~Graph() {
// free the memory
delete [] adj;
}


/**
* @brief add an undirected edge
* @param v node index "from"
* @param w node index "to"
*/
void add_edge(std::size_t v, std::size_t w){
adj[v].push_back(w);
adj[w].push_back(v);
}


/**
```

```
* @brief Method to get the islands of a graph
*/
std::vector<std::vector<std::size_t>> connectedComponents(){

// storage structure for the islands
std::vector<std::vector<std::size_t>> islands;

// set the island index
std::size_t island_idx = 0;

// Mark all the vertices as not visited
bool *visited = new bool[V];
for(std::size_t v = 0; v < V; v++)
visited[v] = false;

// go though all the vertices...
for (std::size_t v = 0; v < V; v++) {

// if v has not been visited...
if (visited[v] == false) {

// add new island, because the recursive process
// has already visited all the island connected to v
if (island_idx >= islands.size())
islands.push_back(std::vector<std::size_t>());

// store all reachable vertices from v recursivelly
DFSUtil(v, visited, &islands, island_idx);

// increase the islands index
island_idx++;
}

}

// sort the islands to maintain raccord
for (std::size_t i=0; i < islands.size(); i++)
std::sort(islands[i].begin(), islands[i].end());

return islands;
}

};

}
```

8.2 Phases reduction

In a three phase system, there may be branches that only transport one or two of the phases. Such systems are unbalanced by design and require a special treatment to prepare for the simulation. As irregular as these systems may be, they exist and a complete simulation engine should be prepared to simulate them too.

To prepare the system for the simulation, we must identify the phases that arrive to each bus under the assumption that all the buses are á-priori able to transport three phases.

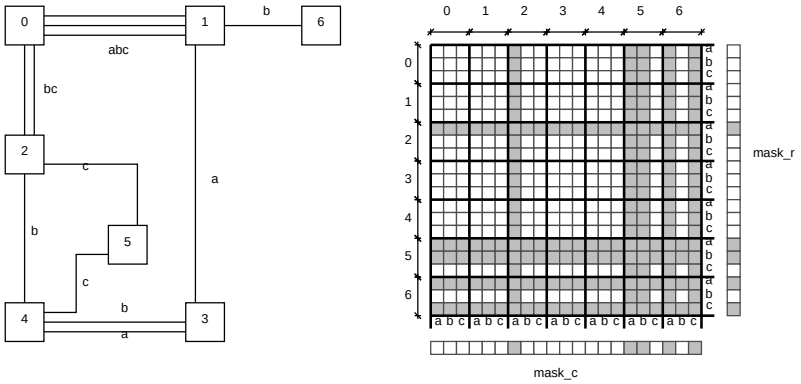


Figure 8.2: This picture shows a 7-node grid with unbalanced connections. The connectivity bus incidence matrix reduction is depicted as well in order to illustrate the phase reduction mechanism that allows the calculation of this circuit. Only the white cells remain after removing the dark cells that correspond to not connected phases.

The algorithms presented in the next sections are prepared to form the full three-phase matrices and vectors and take into account the formation of the mask vectors to be able to reduce the vectors and matrices later.

8.3 Calculation of the voltage, power and current vectors

All numerical methods of electrical calculation need the admittance matrix and a series of vectors of compatible dimensions. These vectors are;

- S: Nodal power injections vector in p.u.
- V: Nodal voltages vector.
- I: Nodal current injections vector.
- Y_{zip}: Nodal shunt admittances vector. These are the admittances corresponding the ZIP model (YISV model in this document). These admittances are added to the diagonal of the admittance matrix (Y_{bus}).

These vectors depend on the magnitudes of the elements connected to each bus (or node). Remember that we have decided that the loads, generators and others are stored in each bus element, this allows us to create a function at the bus object level that "compiles" its voltage, current and power and that these are compiled at the circuit level in the corresponding vectors. The algorithm is as follows:

```
n = number of buses in the circuit
```

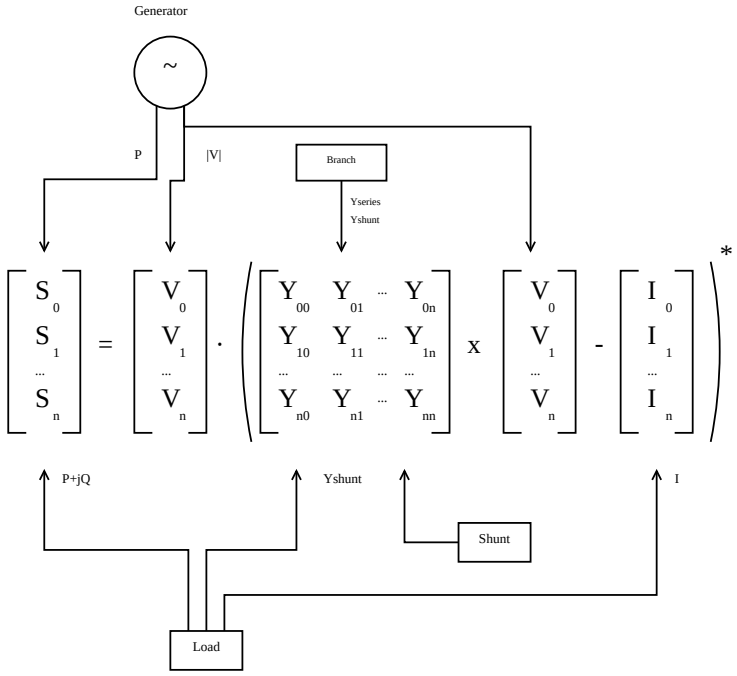


Figure 8.3: Nodal circuit equation. The goal is to compile the properties of the different objects into the calculation matrices and vectors.

```
Y_zip = complex_vector(n)
I = complex_vector(n)
S = complex_vector(n)
V = complex_vector(n)
types = vector(n)

for i=0 to n:
    // assign the values to the vectors at the position i
    buses[i].get_YISV(i, Y, I, S, V, types)
end

// after this loop all the bus related vectors will be filled
// now, we need to divide some of them by the base power to
// have them in per unit.
Y_zip /= Sbase
I /= Sbase
S /= Sbase
```

The last code relies on a function implemented inside each bus object (`get_YISV(i, Y_zip, I, S, V, types)`). The pseudo code of such function would be:

```
i: bus index // external parameter
loads: list of loads within the bus object
generators: list of generators within the bus object
batteries: list of batteries within the bus object

V[i]=1.0

if bus_type is not VD:
```

```
    bus_type=PQ
end

for k=0 to number_of_loads:
    S[i] -= loads[k].S  // loads are a negative injection
    I[i] -= loads[k].I
    Y_zip[i] += loads[k].Y  // the impedance is positive always
end

for k=0 to number_of_generators:
    S[i] += generator[k].P + 0j
    V[i] *= generator[k].V_set
    if bus_type is not VD:
        bus_type=PV
    end
end

for k=0 to number_of_batteries:
    S[i] += batteries[k].P + 0j
    V[i] *= batteries[k].V_set
    if bus_type is not VD:
        bus_type=PV
    end
end

types[i] = bus_type
```

8.4 Calculation of the admittance matrices

For the computations ahead we will need to have a number of admittance-based matrices. Remember that these matrices are only valid for island circuits, this is a multi-island circuit will have a set of these matrices per island. Otherwise the matrices would be singular.

- n: Number of buses.
- m: Number of branches.
- Y_{bus} or simply Y: Complete admittance matrix.

  It is a sparse matrix of size $n \times n$
- Y_{series}: Admittance matrix of the series elements. It contains no value coming from shunt elements or the shunt parts of the branch model.

  It is a sparse matrix of size $n \times n$
- Y_{shunt}: Admittance vector of the shunt elements and the shunt parts of the branch model.

  It is a vector of size n
- Y_f: Admittance matrix of the branches with their *from* bus.

  It is a sparse matrix of size $m \times n$
- C_t: Connectivity matrix of the branches with their *to* bus.
- C_f: Connectivity matrix of the branches with their *from* bus.

  It is a sparse matrix of size $m \times n$
- Y_t: Admittance matrix of the branches with their *to* bus.

  It is a sparse matrix of size $m \times n$

  Where n is the number of buses and m is the number of branches.
  The relation between the admittance matrix and the series and shunt admittance matrices is the following:

$$Y_{bus} = Y_{series} + Y_{shunt} \tag{8.1}$$

Algorithm The algorithmic logic to build the matrices in pseudo code for the positive sequence model is the following:

```
n = number of buses in the circuit
m = number of branches in the circuit

// declare the matrices
Y = complex_sparse(n, n)
Yf = complex_sparse(m, n)
Yt = complex_sparse(m, n)
yshunt = complex_vector(n)
yseries = complex_sparse(n, n)
```

```
// set the buses admittances
for i=0 to n:

    // get all the bus-connected power, current andmittance and voltage
    y, curr, s, v = buses[i].get_YISV()

    Yshunt[i] = y
end

//set the branches admittances
for i=0 to m:
    // get the bus indices matching the busses of this branch
    f = get_bus_index(branches[i].bus_from)
    t = get_bus_index(branches[i].bus_to)

    // compose the basic elements (only for pisitive sequence)
    z_series = complex(branches[i].R, branches[i].X)
    y_shunt = complex(branches[i].G, branches[i].B)
    tap = branches[i].tap_module * exp(-1j * branches[i].angle)
    Ysh = y_shunt / 2
    Ys = 1 / z_series

    // compose the matrix sub elements
    Ytt = Ys + Ysh
    Yff = Ytt / (tap * conj(tap))
    Yft = - Ys / conj(tap)
    Ytf = - Ys / tap
    Yff_sh = Ysh
    Ytt_sh = Yff_sh / (tap * conj(tap))

    // Full admittance matrix
    Ybus[f, f] += Yff
    Ybus[f, t] += Yft
    Ybus[t, f] += Ytf
    Ybus[t, t] += Ytt

    // Y-from and Y-to for the lines power flow computation (optional)
    Yf[i, f] += Yff
    Yf[i, t] += Yft
    Yt[i, f] += Ytf
    Yt[i, t] += Ytt

    // Connectivity matrices
    Cf[i, f] = 1
    Cf[i, t] = -1
    Ct[i, f] = -1
    Ct[i, t] = 1

    // Y shunt
```

```
    Yshunt[f] += Yff_sh
    Yshunt[t] += Ytt_sh

    // Y series
    Yseries[f, f] += Ys / (tap * conj(tap))
    Yseries[f, t] += Yft
    Yseries[t, f] += Ytf
    Yseries[t, t] += Ys
end
```

Algorithm The algorithmic logic to build the matrices in pseudo code for the three-phase model is the following:

```
    For every branch:

    Get the branch submatrices Yff, Yft, Ytf and Ytt.

    a = 0
    for i in phases_from:
        b = 0
        for j in phases_to:

        # Admittance matrix
        Ybus[f * n_phase + i, f * n_phase + j] += Yff[a, b]
        Ybus[f * n_phase + i, t * n_phase + j] += Yft[a, b]
        Ybus[t * n_phase + i, f * n_phase + j] += Ytf[a, b]
        Ybus[t * n_phase + i, t * n_phase + j] += Ytt[a, b]

        # branch-bus admittance matrices (optional)
        Yf[k * n_phase + i, f * n_phase + j] += Yff[a, b]
        Yf[k * n_phase + i, t * n_phase + j] += Yft[a, b]
        Yt[k * n_phase + i, f * n_phase + j] += Ytf[a, b]
        Yt[k * n_phase + i, t * n_phase + j] += Ytt[a, b]

        # branch-bus connectivity matrices
        Cf[k * n_phase + i, f * n_phase + j] = 1
        Cf[k * n_phase + i, t * n_phase + j] = -1
        Ct[k * n_phase + i, f * n_phase + j] = -1
        Ct[k * n_phase + i, t * n_phase + j] = 1

        # masks
        mask_r[f * n_phase + i] = 1
        mask_r[t * n_phase + i] = 1

        mask_c[f * n_phase + j] = 1
        mask_c[t * n_phase + j] = 1

        mask_k[k * n_phase + i] = 1
```

```
        b += 1
    end

        a += 1
    end
```

This algorithm will produce the full three-phase matrices. These matrices need to be reduced as explained in the section 8.2.

9 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:

$$S = V \times (Y \times V - I)^* \tag{9.1}$$

Where:

- S: Vector of power injections.
- I: Vector of current injections.
- V: Vector of voltages.
- Y: Admittance matrix.

Note that the equation 9.1 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.

9.1 Types of buses

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

- $|V|$: Voltage module.
- δ: 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	$\|V\|$	δ	P	Q
PQ	Calc.	Calc.	Set	Set
PV	Set	Calc.	Set	Calc.
VD (slack)	Set	Set	Calc.	Calc.

Table 9.1: Bus types.

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

9.2 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:

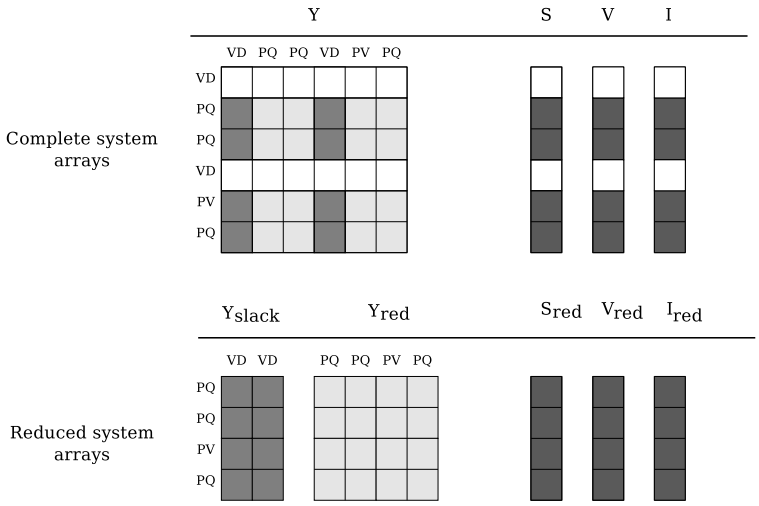


Figure 9.1: Circuit's admittance matrix and vectors reduction. Representation for a six node circuit with two slack nodes.

$$Y_{red} = Y[pqpv, pqpv] \tag{9.2}$$

$$Y_{slack} = Y[pqpv, vd] \tag{9.3}$$

$$V_{red} = V[pqpv] \tag{9.4}$$

$$V_{slack} = V[vd] \tag{9.5}$$

$$S_{red} = S[pqpv] \tag{9.6}$$

$$I_{red} = I[pqpv] \tag{9.7}$$

Once we have all the reduced magnitudes, it means that we have removed the slack (VD) nodes from the circuit, but we must keep their influence in the rest of the nodes in the form of current injections.

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

$$V_{prev} = V_{red} \tag{9.8}$$

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

$$I_{slack} = Y_{slack} \times V_{slack} \tag{9.9}$$

The voltages that arise from the slack current injections are:

$$C_k = Y_{red} \times I_{slack} \tag{9.10}$$

The total injected currents including the power injections are:

$$I_k = \frac{S_{red}}{V_{prev}} + I_{red} \tag{9.11}$$

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

$$V_k = Y_{red}^{-1} \times I_k - C_k \tag{9.12}$$

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

$$error = ||V_{prev} - V_k||_\infty = max(abs(V_{prev} - V_k)) \tag{9.13}$$

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.

$$V_k[pv_{red}] = V_k \cdot \frac{|V^{esp}[pv_{red}]|}{|V_k[pv_{red}]|} \tag{9.14}$$

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:

$$V_{prev} = V_k \tag{9.15}$$

Repeat equations 9.11 to 9.15 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.

$$V[pqpv] = V_k \tag{9.16}$$

9.3 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 [9.1] 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.

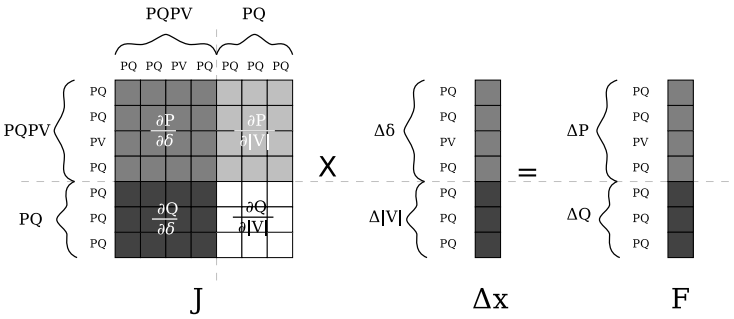


Figure 9.2: Jacobian based linear system to compute the voltage increments.

The Jacobian linear system ($J \times \Delta x = F$), allows the computation of the voltage increment (Δx) to apply to the voltage solution (V) in order to achieve a voltage that solves the power flow equation 9.1. 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.

9.3.1 Building the Jacobian matrix

The Jacobian matrix (J) is the derivative of the fundamental power flow equation 9.1, with respect to the voltage magnitudes to be solved $|V|$ and δ. 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.

$$I_{diag} = diag(Y \times V - I) \tag{9.17}$$

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

$$V_{diag} = diag(V) \tag{9.18}$$

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

$$E_{diag} = diag(V/|V|) \tag{9.19}$$

Now, Compute the derivative of the power injections S with respect to the voltage module $|V|$:

$$\frac{\partial S}{\partial |V|} = V_{diag} \times (Y \times E_{diag})^* + I^*_{diag} \times E_{diag} \tag{9.20}$$

Compute the derivative of the power injections S with respect to the voltage angle δ:

$$\frac{\partial S}{\partial \delta} = 1j \cdot V_{diag} \times (I_{diag} - Y \times V_{diag})^* \tag{9.21}$$

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 9.2.

$$J = \begin{bmatrix} Re\left(\frac{\partial S}{\partial \delta}[pqpv, pqpv]\right) & Re\left(\frac{\partial S}{\partial |V|}[pqpv, pq]\right) \\ Im\left(\frac{\partial S}{\partial \delta}[pq, pqpv]\right) & Im\left(\frac{\partial S}{\partial |V|}[pq, pq]\right) \end{bmatrix} \tag{9.22}$$

Observe that the Jacobian matrix is composed of four subsets of the previously computed derivatives $\frac{\partial S}{\partial \delta}$ and $\frac{\partial S}{\partial |V|}$, where combinations of the PQ and PV node indices are selected.

$pqpv$ 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.

9.3.2 Newton-Raphson

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:

$$\Delta S = S_{specified} - S_{calc} \tag{9.23}$$

Where:

$$s_{calc} = V \times (Y \times V - I_{bus})^* \tag{9.24}$$

The minimization of ΔS occurs via se recursive solution of the following linear system:

$$J \times \Delta x = F \tag{9.25}$$

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

$$F = \begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} \tag{9.26}$$

$$\Delta Q = Im(\Delta S[pq]) \tag{9.27}$$

$$\Delta P = Re(\Delta S[pqpv]) \tag{9.28}$$

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).

$$error = ||F||_\infty = max(abs(F)) \tag{9.29}$$

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 $J \times \Delta x = F$.

$$\Delta x = J^{-1} \times F \tag{9.30}$$

Notice that Δx is a vector with the voltage angles (δ) for the PQ and PV nodes followed by the voltage modules ($|V|$) 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 $\Delta|V|$ and $\Delta\delta$, 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.

$$\Delta\delta[pqpv[i]] = \Delta x[i] \quad \forall i \in 0..npqpv \tag{9.31}$$

$$\Delta|V|[pq[i]] = \Delta x[i + npqpv] \quad \forall i \in 0..npq \tag{9.32}$$

Then we need to add those values to the voltage module and angle vectors $|V|$ and δ:

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

$$|V|^{(k+1} = |V|^{(k} + \Delta|V|^{(k} \tag{9.33}$$

$$\delta^{(k+1} = \delta^{(k} + \Delta\delta^{(k} \tag{9.34}$$

At last, we convert the polar vectors $|V|$ and δ into a single complex voltage vector V.

$$V^{(k+1} = |V|^{(k+1} \cdot \left(cos(\delta^{(k+1}) + 1j \cdot sin(\delta^{(k+1})\right) \tag{9.35}$$

The method consists in repeating formulas 9.24 to 9.35 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 mismatch function (F) using the initial voltage solution (V). Equation 9.26.
3. Compute the error. Equation 9.29.
4. While $error > tolerance$ or $iterations < max\_iterations$:
   (a) Compute the Jacobian
   (b) Solve the linear system. Equation 9.30.
   (c) Assign Δx to V. Equations 9.31 to 9.35.
   (d) Compute the mismatch function (F) using the latest voltage solution (V). Equation 9.26.
   (e) Compute the error. Equation 9.29.
   (f) $iterations = iterations + 1$
5. End.

9.3.3 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:

$$A = J^\top \times J + \lambda \cdot I \tag{9.36}$$

Where λ is computed only in the first iteration as:

$$\lambda = 0.001 \cdot max(diag(J \times J^\top)) \tag{9.37}$$

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

F is provided in the equation 9.26.

$$rhs = J^\top \times F \tag{9.38}$$

Solve the voltage increments vector Δx. 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 9.2.

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

$$\Delta x = A^{-1} \times rhs \tag{9.39}$$

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

$$f = \frac{1}{2} \cdot (F \times F^\top) \tag{9.40}$$

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.

$$\rho = \frac{f^{(k-1} - f^{(k}}{\frac{1}{2} \cdot (\Delta x \times (\lambda \cdot \Delta x + rhs)^\top)} \tag{9.41}$$

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 9.31 to 9.35. We also need to modify λ:

$$\lambda = \lambda \cdot max(\frac{1}{3}, 1 - (2 \cdot \rho - 1)^3) \tag{9.42}$$

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 function (F) using the initial voltage solution (V). Equation 9.26.
3. Compute the error. Equation 9.29.
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 9.36.
   (b) Compute the linear system right hand side using equation 9.38.
   (c) Solve the linear system to obtain Δx. Equation 9.39.
   (d) Compute the objective function f using equation 9.40.
   (e) Compute the decision value ρ using equation 9.41.
   (f) If $\rho > 0$:
      i. Assign Δx to V. Equations 9.31 to 9.35.
      ii. Update λ using equation 9.42.
      iii. Set $\nu = 2$.
      iv. Set the Jacobian update flag to true.
   (g) If $\rho \leq 0$:
      i. Assign Δx to V. Equations 9.31 to 9.35.
      ii. Update $\lambda = \nu \cdot \lambda$.
      iii. Update $\nu = 2 \cdot \nu$.
      iv. Set the Jacobian update flag to false.
   (h) Compute the mismatch function (F) using the latest voltage solution (V). Equation 9.26.
   (i) Compute the error. Equation 9.29.
   (j) $iterations = iterations + 1$
5. End.

9.4 Linear DC 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, $|v| = 1$ for load nodes and $|v| = v_{set}$ 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 ($\mathbf{Y}_{red}$) is obtained by removing the rows and columns corresponding to the slack nodes. Likewise the removed elements conform the ($\mathbf{Y}_{slack}$) matrix.

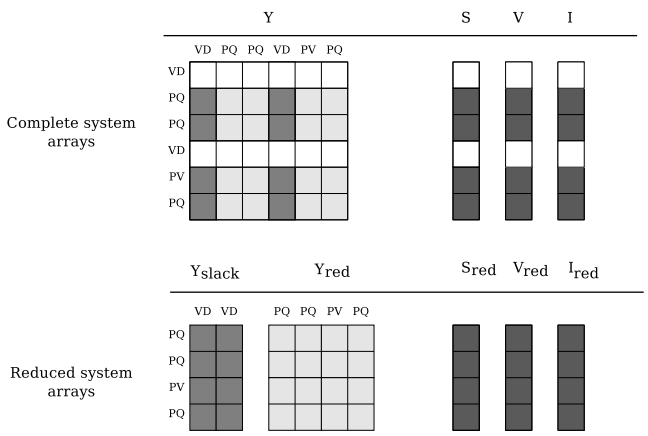


Figure 9.3: Matrix reduction (VD: Slack, PV: Voltage controlled, PQ: Power controlled)

$$\mathbf{P} = Re(\mathbf{S}_{red}) + (-Im(\mathbf{Y}_{slack}) \cdot angle(\mathbf{V}_{slack}) + Re(\mathbf{I}_{red})) \cdot |\mathbf{V}_{red}| \quad (9.43)$$

The equation 9.43 computes the DC power injections as the sum of the different factors mentioned:

1. $Re(\mathbf{S}_{red})$: Real part of the reduced power injections.
2. $Im(\mathbf{Y}_{slack}) \cdot angle(\mathbf{V}_{slack}) \cdot |v_{red}|$: Currents that appear by removing the slack nodes while keeping their influence, multiplied by the voltage module to obtain power.
3. $Re(\mathbf{I}_{red}) \cdot |v_{red}|$: Real part of the grid reduced current injections, multiplied by the voltage module to obtain power.

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

$$\mathbf{V}_{angles} = Im(\mathbf{Y}_{red})^{-1} \times \mathbf{P} \quad (9.44)$$

The new voltage is then:

$$\mathbf{V}_{red} = |\mathbf{V}_{red}| \cdot e^{1j \cdot \mathbf{V}_{angles}} \quad (9.45)$$

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.

9.5 Linear AC power flow

The AC linear approximation 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 [2], we obtain a way to solve circuits in

[2] 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.

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} \Delta\theta \\ \Delta|V| \end{bmatrix} = \begin{bmatrix} Rhs_1 \\ Rhs_2 \end{bmatrix} \tag{9.46}$$

Where:

- $A_{11} = -Im\left(Y_{series}[pqpv, pqpv]\right)$
- $A_{12} = Re\left(Y_{bus}[pqpv, pq]\right)$
- $A_{21} = -Im\left(Y_{series}[pq, pqpv]\right)$
- $A_{22} = -Re\left(Y_{bus}[pq, pq]\right)$
- $Rhs_1 = Re(S[pqpv])$
- $Rhs_2 = Im(S[pq])$

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

9.6 Holomorphic embedding

First introduced by Antonio Trias in 2012 [3], 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) [4]. 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.

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

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

9.6.1 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

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.

9.6.2 *Fundamentals*

The fundamental equation that defines the power flow problem is:

$$\mathbf{S} = \mathbf{V} \times (\mathbf{Y} \times \mathbf{V})^* \tag{9.47}$$

Most usefully represented like this:

$$\mathbf{Y} \times \mathbf{V} = \left(\frac{\mathbf{S}}{\mathbf{V}}\right)^* \tag{9.48}$$

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 9.48 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).

$$\mathbf{Y}_{series} \times \mathbf{V} = \left(\frac{\mathbf{S}}{\mathbf{V}}\right)^* - \mathbf{Y}_{shunt}\mathbf{V} \tag{9.49}$$

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:

$$\mathbf{Y} \times \mathbf{V}(\alpha) = \left(\frac{\alpha \mathbf{S}}{\mathbf{V}(\alpha)}\right)^* - \alpha \mathbf{Y}_{shunt} \times \mathbf{V}(\alpha) = \frac{\alpha \mathbf{S}^*}{\mathbf{V}(\alpha)^*} - \alpha \mathbf{Y}_{shunt} \mathbf{V}(\alpha) \quad (9.50)$$

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 $\mathbf{V}(\alpha)$.
- Regarding the α that multiplies **S**, the amount of power (α**S**) 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 α $\mathbf{Y}_{shunt}$ is to provoke the first voltage coefficients to be one. $\mathbf{Y}_{series} \times \mathbf{V}[0] = 0$, 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:

$$V(\alpha) = \sum_n^\infty V_n \alpha^n \quad (9.51)$$

9.6.3 Holomorphicity check

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

$$\partial \left(\mathbf{V}(\alpha)^*\right) / \partial \alpha^* = \partial \left(\sum_n^\infty V_n^* (\alpha^n)^*\right) / \partial \alpha^* = \sum_n^\infty \alpha^n V_n^* (\alpha^{n-1})^* \quad (9.52)$$

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

$$\partial \left(\mathbf{V}(\alpha^*)\right)^* / \partial \alpha^* = \partial \left(\sum_n^\infty \mathbf{V}_n^* \alpha^n\right) / \partial \alpha^* = 0 \quad (9.53)$$

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:

$$\mathbf{Y}_{series} \times \mathbf{V}(\alpha) = \frac{\alpha \mathbf{S}^*}{\mathbf{V}^*(\alpha^*)} - \alpha \mathbf{Y}_{shunt} \mathbf{V}(\alpha) \quad (9.54)$$

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

$$\frac{1}{\mathbf{V}(\alpha)} = \mathbf{W}(\alpha) \longrightarrow \mathbf{W}(\alpha)\mathbf{V}(\alpha) = 1 \longrightarrow \sum_{n=0}^\infty \mathbf{W}_n \alpha^n \sum_{n=0}^\infty \mathbf{V}_n \alpha^n = 1 \quad (9.55)$$

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

$$\mathbf{W}_n = \begin{cases} \dfrac{1}{\mathbf{V}_0}, & n = 0 \\ -\dfrac{\sum_{k=0}^{n} \mathbf{W}_k \mathbf{V}_{n-k}}{\mathbf{V}_0}, & n > 0 \end{cases} \tag{9.56}$$

Now, the equation 9.54 is:

$$\mathbf{Y}_{series} \times \mathbf{V}(\alpha) = \alpha \mathbf{S}^* \cdot \mathbf{W}(\alpha)^* - \alpha \mathbf{Y}_{shunt} \mathbf{V}(\alpha) \tag{9.57}$$

Substituting the series by their McLaurin expressions:

$$\mathbf{Y}_{series} \times \sum_{n=0}^{\infty} \mathbf{V}_n \alpha^n = \alpha \mathbf{S}^* \left(\sum_{n=0}^{\infty} \mathbf{W}_n \alpha^n \right)^* - \alpha \mathbf{Y}_{shunt} \sum_{n=0}^{\infty} \mathbf{V}_n \alpha^n \tag{9.58}$$

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

$$\mathbf{V}_n = \begin{cases} 0, & n = 0 \\ \mathbf{S}^* \mathbf{W}_{n-1}^* - Y_{shunt} \mathbf{V}_{n-1}, & n > 0 \end{cases} \tag{9.59}$$

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 9.4. For instance $\mathbf{V}_n$ is the n^{th} row of the coefficients structure **V**.

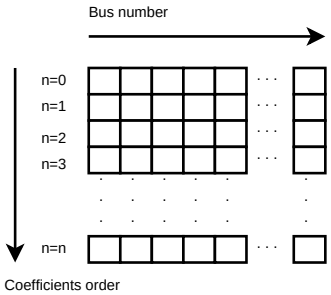


Figure 9.4: Structure of the coefficients

9.6.4 *Padé approximation*

The equation 9.51 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 9.51 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 9.51 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 $\mathbf{V}(\alpha)$, 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:

$$Voltage\_value\_approximation = \frac{P_N(\alpha)}{Q_M(\alpha)} \quad \forall \alpha \in [0, 1] \tag{9.60}$$

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!)

$$P_N(\alpha) = p_0 + p_1\alpha + p_2\alpha^2 + ... + p_N\alpha^N \tag{9.61}$$

$$Q_M(\alpha) = 1 + q_1\alpha + q_2\alpha^2 + ... + q_M\alpha^M \tag{9.62}$$

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.

$$\begin{gathered} q_M V_{N-M+1} + q_{M-1} V_{N-M+2} + ... + q_1 V_N = 0 \\ q_M V_{N-M+2} + q_{M-1} V_{N-M+3} + ... + q_1 V_{N+1} = 0 \\ ... \\ q_M V_N + q_{M-1} V_{N+1} + ... + q_1 V_{N+M+1} + V_{N+M} = 0 \end{gathered} \tag{9.63}$$

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

$$\begin{gathered} V_0 - p_0 = 0 \\ q_1 V_0 + V_1 p_1 = 0 \\ q_2 V_0 + q_1 V_1 + V_2 - p_2 = 0 \\ q_3 V_0 + q_2 V_1 + q_1 V_2 + V_3 - p_3 = 0 \\ ... \\ q_M V_{N-M} + q_{M-1} V_{N-M+1} + ... + + V_N - p_N = 0 \end{gathered} \tag{9.64}$$

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 9.60 for $\alpha = 1$.

This process is done for every column of coefficients $\mathbf{V} = \{V_0, V_1, V_2, V_3, ..., V_n\}$ of the structure depicted in the figure 9.4. 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.

9.6.5 *Formulation with PV nodes*

The section 9.6.2 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 [5], which does not work as published, hence the correction.

[5] 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 9.6.2.

For Slack nodes:

$$V(\alpha) = V^{SP} \quad \forall \alpha = 0 \tag{9.65}$$

For PQ nodes:

$$\begin{cases} \mathbf{Y} \times \mathbf{V}(\alpha) = 0 & \forall \alpha = 0 \\ \mathbf{Y} \times \mathbf{V}(\alpha) = \dfrac{\alpha \mathbf{S}}{\mathbf{V}^*(\alpha^*)} & \forall \alpha > 0 \end{cases} \tag{9.66}$$

For PV nodes:

$$\begin{cases} \mathbf{Y} \times \mathbf{V}(\alpha) = \dfrac{\mathbf{S}}{\mathbf{V}^*(\alpha^*)} & \forall \alpha = 0 \\ \mathbf{Y} \times \mathbf{V}(\alpha) = \dfrac{\mathbf{S} - j\mathbf{Q}(\alpha)}{\mathbf{V}^*(\alpha^*)} & \forall \alpha > 0 \end{cases} \tag{9.67}$$

$$\begin{cases} V(\alpha)V^*(\alpha^*) = |V_0|^2 & \forall \alpha = 0 \\ V(\alpha)V^*(\alpha^*) = |V_0|^2 + (|V^{SP}|^2 - |V_0|^2) & \forall \alpha > 0 \end{cases} \tag{9.68}$$

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 [6].

[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

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

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ G_{21} & -B_{21} & G_{22} & -B_{22} & G_{23} & -B_{23} \\ B_{21} & G_{21} & B_{22} & G_{22} & B_{23} & G_{23} \\ G_{31} & -B_{31} & G_{32} & -B_{32} & G_{33} & -B_{33} \\ B_{31} & G_{31} & B_{32} & G_{32} & B_{33} & G_{33} \end{bmatrix} \times \begin{bmatrix} V[n]_{re,1} \\ V[n]_{im,1} \\ V[n]_{re,2} \\ V[n]_{im,2} \\ V[n]_{re,3} \\ V[n]_{im,3} \end{bmatrix} = \begin{bmatrix} V^{SP}_{re,1} \\ V^{SP}_{im,1} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \forall n = 0 \tag{9.69}$$

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

$$\mathbf{S} = \mathbf{V}[0] \cdot (\mathbf{Y}_{bus} \times \mathbf{V}[0])^* \tag{9.70}$$

$$\mathbf{Q}_i[0] = Im(\mathbf{S}_i) \quad \forall i \in PV \tag{9.71}$$

The initial inverse voltage coefficients $\mathbf{W}[0]$ are obtained by:

$$W_i[0] = \frac{1}{V_i[0]} \quad \forall i \in N \tag{9.72}$$

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.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ G_{21} & -B_{21} & G_{22} & -B_{22} & G_{23} & -B_{23} & W[0]_{im} \\ B_{21} & G_{21} & B_{22} & G_{22} & B_{23} & G_{23} & W[0]_{re} \\ G_{31} & -B_{31} & G_{32} & -B_{32} & G_{33} & -B_{33} & 0 \\ B_{31} & G_{31} & B_{32} & G_{32} & B_{33} & G_{33} & 0 \\ 0 & 0 & V[0]_{re} & V[0]_{im} & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} V[n]_{re,1} \\ V[n]_{im,1} \\ V[n]_{re,2} \\ V[n]_{im,2} \\ V[n]_{re,3} \\ V[n]_{im,3} \\ Q_2[n] \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ f2_{re} \\ f2_{im} \\ f1_{re} \\ f1_{im} \\ \epsilon[n] \end{bmatrix} \quad \forall n > 0 \tag{9.73}$$

Where:

$$f1 = S_i^* \cdot W_i^*[n-1] \quad \forall i \in PQ \tag{9.74}$$

$$f2 = P_i \cdot W_i^*[n-1] + conv(n, Q_i, W_i^*) \quad \forall i \in PV \tag{9.75}$$

$$\epsilon[n] = \delta_{n1} \cdot \frac{1}{2}\left(|V_i^S P|^2 - |V_i[0]|^2\right) - \frac{1}{2} conv(n, V_i, V_i^*) \quad \forall i \in PV, n > 0 \tag{9.76}$$

The convolution $conv$ is defined as:

$$conv(n, A, B) = \sum_{m=0}^{n-1} A[m] \cdot B[n-m] \tag{9.77}$$

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 $\mathbf{V}[n]$ and the reactive power at the PV nodes $Q[n]$ is obtained solving the linear system (eq 9.73), we must solve the inverse voltage coefficients of order n for all the buses:

$$W_i[n] = \frac{-\sum_{m=0}^{n} W_i[m] \cdot V_i[n-m]}{V_i[0]} \quad \forall i \in N, n > 0 \tag{9.78}$$

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 9.70) with the specified power injections $\mathbf{S}^{SP}$:

$$mismatch = \mathbf{S} - \mathbf{S}^{SP} \tag{9.79}$$

$$error = |mismatch|_\infty = max(abs(mismatch)) \tag{9.80}$$

9.7 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:

$$\mathbf{I}_f = (\mathbf{C}_f \times \mathbf{Y}_{bus}) \times \mathbf{V} \tag{9.81}$$

$$\mathbf{I}_t = (\mathbf{C}_t \times \mathbf{Y}_{bus}) \times \mathbf{V} \tag{9.82}$$

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

Then we compute the power values:

$$\mathbf{S}_f = (\mathbf{C}_f \times \mathbf{V}) \cdot \mathbf{I}_f^* \tag{9.83}$$

$$\mathbf{S}_t = (\mathbf{C}_t \times \mathbf{V}) \cdot \mathbf{I}_t^* \tag{9.84}$$

These are element-wise multiplications, 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:

$$\textbf{losses} = |\mathbf{S}_f - \mathbf{S}_t| \cdot Sbase \tag{9.85}$$

And also the branches loading in per unit as:

$$\textbf{loading} = \frac{max(|\mathbf{S}_f|, |\mathbf{S}_t|) \cdot Sbase}{\textbf{rate}} \tag{9.86}$$

The variables are:

- $\mathbf{Y}_{bus}$: Bus admittance matrix in p.u. See 8.4.
- $\mathbf{C}_f$: Connectivity matrix of the branches and the *from* buses.
- $\mathbf{C}_t$:Connectivity matrix of the branches and the *to* buses.
- **V**: Array of bus voltages in p.u.
- $\mathbf{I}_f$: Array of currents at the *from* buses in p.u.
- $\mathbf{I}_t$: Array of currents at the *to* buses in p.u.
- $\mathbf{S}_f$: Array of powers at the *from* buses in p.u.
- $\mathbf{S}_t$: Array of powers at the *to* buses in p.u.
- **rate**: Array of branch ratings in MVA.

9.8 Reactive power control: The outer loop

The power flow result computes values of reactive power for the PV nodes. 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 [7] 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 $V_i \geq Vset_i$, then it remains a PQ bus at current iteration and set $Q_i = Qmin_i$.
   - If $V_i < Vset_i$, 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 $V_i = Vset_i$.
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 $V_i \leq Vset_i$, then the bus i still a PQ bus and set $Q_i = Qmax_i$.
   - If $V_i > Vset_i$, 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 $V_i = Vset_i$
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.

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.

[7] 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

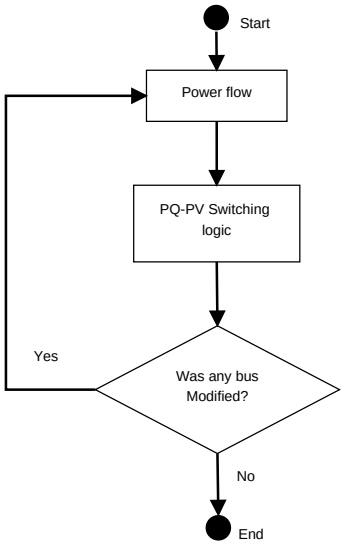


Figure 9.5: $PQ - PV$ switching algorithm.

10 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.

10.1 DC 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:

$$S_i = V_i \sum_{j}^{nodes} \left(Y_{ij} \cdot V_j\right)^* \quad \forall i \in nodes \tag{10.1}$$

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 formulation developed in the previous chapter in order to accommodate the generators dispatch.

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

$$min: \sum_{i}^{Generators+Slack} cost_i \cdot PG_i \tag{10.2}$$

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

$$PG_i^{min} \leq PG_i \leq PG_i^{max} \tag{10.3}$$

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

$$PG_i - PD_i = \sum_{j}^{PQPV} B_{ij} \cdot \theta_j \quad \forall i \in PQPV \tag{10.4}$$

The slack voltage angles are set to zero.

$$\theta_k = 0 \quad \forall k \in Slack \tag{10.5}$$

The branch flows must be within limits.

$$-F_{ij}^{max} \leq B_{ij}(\theta_i - \theta_j) \leq F_{ij}^{max} \tag{10.6}$$

This is reformulated as:

$$B_{ij}(\theta_i - \theta_j) \leq F_{ij}^{max} \tag{10.7}$$

$$B_{ij}(\theta_j - \theta_i) \leq F_{ij}^{max} \tag{10.8}$$

Set the slack generator power:

$$PG_i - PD_i = \sum_{j}^{nodes} B_{ij} \cdot \theta_j \quad \forall i \in Slack \tag{10.9}$$

- 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

10.2 AC optimal power flow

This section introduces the use of the AC linear power flow (see 9.5) as theoretical source to obtain an AC optimal power flow. 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:

$$min: \sum_{i}^{Generators} cost_i \cdot PG_i \quad \forall i \in generators \tag{10.10}$$

ST:

$$PG_i^{min} \leq PG_i \leq PG_i^{max} \quad \forall i \in generators \tag{10.11}$$

$$\theta_i^{min} \leq \theta_i \leq \theta_i^{max} \quad \forall i \in pvpq \tag{10.12}$$

$$|V|_i^{min} \leq |V|_i \leq |V|_i^{max} \quad \forall i \in pq \tag{10.13}$$

The equation 9.46 can be transformed to the following equations:

$$-\sum_{k}^{pvpq} Bs_{k,i} \cdot \Delta\theta_i + \sum_{k}^{pq} G_{k,i} \cdot \Delta|V_i| = \sum_{g}^{generators_i} PG_g - \sum_{l}^{loads_i} PD_l \quad \forall i \in pvpq \tag{10.14}$$

$$-\sum_{k}^{pqpv} Gs_{k,i} \cdot \Delta\theta_i - \sum_{k}^{pq} B_{k,i} \cdot \Delta|V_i| = -\sum_{l}^{loads_i} QD_l \quad \forall i \in pq \tag{10.15}$$

$$\theta_i = \Delta\theta_i \tag{10.16}$$

$$|V_i| = |V_i|^{SP} + \Delta|V_i| \tag{10.17}$$

11 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.

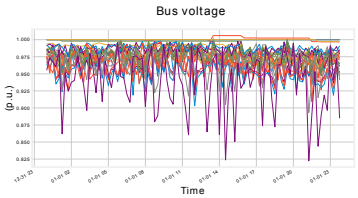


Figure 11.1: Example of voltage results for a time series simulation.

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.

12 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.

12.1 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.

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

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

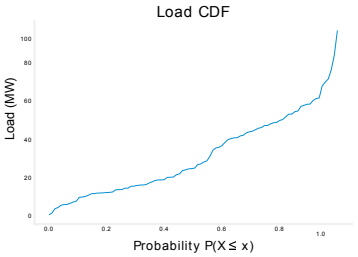


Figure 12.1: Example of cumulative distribution function of a load.

12.2 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 12.3.

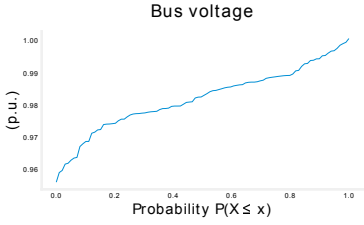


Figure 12.2: Example of cumulative distribution function of the voltage. This is the Result of a probabilistic power flow.

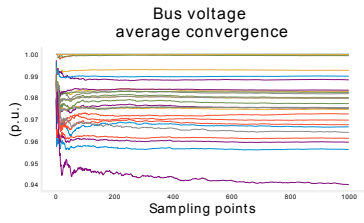


Figure 12.3: Example of voltage convergence. This is the Result of a probabilistic power flow.

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: $iterations = 0$, $\mathbf{V}_{summation} = 0$
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 9) with those arrays and get the voltages vector $\mathbf{V}$.
   (c) Increase the number of iterations: $iterations+ = 1$
   (d) Compute the voltages summation: $\mathbf{V}_{summation}+ = \mathbf{V}$

(e) Compute the voltage average: $\mathbf{V}_{mean} = \mathbf{V}_{summation}/iterations$

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

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

4. End.

13 WLS State estimation

Weighted Least Squares State estimation is a method to correct field measurements with the mathematical grid equilibrium equations (See equation 9.1). In this chapter we are going to follow the implementation suggested in the book from Antonio Gómez Expósito and Ali Abur [1].

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

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

$$\left[H^T \times W \times H + \delta \cdot diag(H^T \times W \times H)\right] \times \Delta x = \left[H^T \times W \times (z - h)\right] \tag{13.1}$$

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 13.1 is solved using the Levenberg-Marquardt method (See 9.3.3) 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.

$$W_{i,i} = \frac{1}{\sigma_i^2} \quad \forall i \in Measurements \tag{13.2}$$

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

$$H = \begin{bmatrix} A1 & A2 \\ B1 & B2 \\ C1 & C2 \\ D1 & D2 \\ E1 & E2 \\ F1 & F2 \end{bmatrix} \tag{13.3}$$

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

$$\frac{\partial S_{inj}}{\partial \theta} = j \cdot V_{diag} \times (I_{bus,diag} - Y_{bus} \times V_{diag})^* \tag{13.4}$$

$$\frac{\partial S_{inj}}{\partial |V|} = j \cdot E_{diag} \times (I_{bus,diag} + Y_{bus} \times V_{diag})^* \tag{13.5}$$

$$\frac{\partial S_{flow}}{\partial \theta} = j \cdot (I^*_{f,diag} \times C_f \times V_{diag} - [C_f \times V]_{diag} \times Y^*_f \times V^*_{diag}) \tag{13.6}$$

$$\frac{\partial S_{flow}}{\partial |V|} = I^*_{f,diag} \times C_f \times E_{diag} - [C_f \times V]_{diag} \times Y^*_f \times E^*_{diag} \tag{13.7}$$

Note: All these derivatives are square matrices in complex numbers.

$$\frac{\partial I_{mag}}{\partial \theta} = j \cdot Y_f \times V_{diag} \tag{13.8}$$

$$\frac{\partial I_{mag}}{\partial |V|} = j \cdot Y_f \times E_{diag} \tag{13.9}$$

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

$$E = \frac{V}{|V|} \tag{13.10}$$

$$I_f = C_f \times Y_{bus} \times V = Y_f \times V \tag{13.11}$$

$$I_{inj} = Y_{bus} \times V + I_{bus} \tag{13.12}$$

$$S_{inj} = V \times I^*_{inj} \tag{13.13}$$

$$S_f = V_{diag} \times I_f^* \tag{13.14}$$

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:

$$A1 = Re\left(\frac{\partial S_{flow}}{\partial |V|}\right)_{[np_{flow}, pqpv]} \tag{13.15}$$

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

$$A2 = Re\left(\frac{\partial S_{flow}}{\partial \theta}\right)_{[np_{flow}, :]} \tag{13.16}$$

$$B1 = Re\left(\frac{\partial S_{inj}}{\partial |V|}\right)_{[np_{inj}, pqpv]} \tag{13.17}$$

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

$$B2 = Re\left(\frac{\partial S_{inj}}{\partial \theta}\right)_{[np_{inj}, :]} \tag{13.18}$$

$$C1 = Im\left(\frac{\partial S_{flow}}{\partial |V|}\right)_{[nq_{flow}, pqpv]} \tag{13.19}$$

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

$$C2 = Im\left(\frac{\partial S_{flow}}{\partial \theta}\right)_{[nq_{flow}, :]} \tag{13.20}$$

$$D1 = Im\left(\frac{\partial S_{inj}}{\partial |V|}\right)_{[nq_{inj}, pqpv]} \tag{13.21}$$

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

$$D2 = Im\left(\frac{\partial S_{inj}}{\partial \theta}\right)_{[nq_{inj}, :]} \tag{13.22}$$

$$E1 = abs\left(\frac{\partial I_{flow}}{\partial |V|}\right)_{[nI_{flow}, pqpv]} \tag{13.23}$$

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

$$E2 = abs\left(\frac{\partial I_{flow}}{\partial \theta}\right)_{[nI_{flow}, :]} \tag{13.24}$$

$$F1 = \frac{\partial |V|}{\partial |V|}_{[nv, pqpv]} = diag([1])_{[nv, pqpv]} \tag{13.25}$$

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

$$F2 = \frac{\partial |V|}{\partial \theta}_{[nv, :]} = [0]_{[nv, :]} \tag{13.26}$$

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.

$$h = \begin{bmatrix} a3 \\ b3 \\ c3 \\ d3 \\ e3 \\ f3 \end{bmatrix} \tag{13.27}$$

$$a3 = Re\left(S_{flow}\right)_{[np_{flow}]} \tag{13.28}$$

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

$$b3 = Re\left(S_{inj}\right)_{[np_{inj}]} \tag{13.29}$$

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

$$c3 = Im\left(S_{flow}\right)_{[nq_{flow}]} \tag{13.30}$$

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

$$d3 = Im\left(S_{inj}\right)_{[nq_{inj}]} \tag{13.31}$$

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

$$e3 = abs\left(I_f\right)_{[nI_{flow}]} \tag{13.32}$$

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

$$f3 = |V|_{[nv]} \tag{13.33}$$

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

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

$$\Delta\mathbf{x} = [\Delta\theta_{pqpv} \quad \Delta|\mathbf{V}|_{all}] \tag{13.34}$$

In this fashion we can split the increments vector:

$$\Delta\theta_{pqpv} = \Delta\mathbf{x}[: npqpv]] \tag{13.35}$$

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

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

$$\Delta|\mathbf{V}| = \Delta\mathbf{x}[npqpv ::]] \tag{13.36}$$

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

$$\mathbf{V}^{(k+1} = \mathbf{V}^{(k} + \Delta|\mathbf{V}|^{-j\cdot\Delta\theta} \tag{13.37}$$

The outcome of the method are the node voltages (vector V). From those one shall compose the branch power flows using the equations described in 9.7.

13.0.1 *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:

$$\begin{bmatrix} \frac{R}{\alpha} & H & 0 \\ H^T & 0 & C^T \\ 0 & C & 0 \end{bmatrix} \times \begin{bmatrix} \mu \\ \Delta x \\ \lambda \end{bmatrix} = \begin{bmatrix} z-h \\ 0 \\ -c(x) \end{bmatrix} \tag{13.38}$$

Where:

- H: Jacobian matrix for the estate estimation problem of the measurements with uncertainty. (See 13.3)
- C: Jacobian matrix for the estate estimation problem of the measurements without uncertainty. (See 13.3)
- R: Matrix of standard deviations. It is the inverse of the weights diagonal matrix.(See 13.2)
- 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 13.27)
- $c(x)$: calculated magnitudes matching the measurements structure for the measurements without uncertainty.

Conditioning value (α)

$$\alpha = \frac{1}{max(W_{i,i})} \tag{13.39}$$

Matrix of standard deviations (R)

$$R_{i,i} = \sigma_{i,i}^2 \tag{13.40}$$

14 Short-circuit

15 Transient stability

Bibliography

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.

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.

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.