Power Flow Calculation Methods: A Comprehensive Review

Leonard Schulte, Albert Moser

IAEW at RWTH Aachen University, l.schulte@iaew.rwth-aachen.de,
www.iaew.rwth-aachen.de

Kurzfassung/Abstract: Power flow calculation in symmetrical 3-phase AC systems is one of the most fundamental tools in the analysis of electrical power systems. The purpose of this paper is to provide a comprehensive overview of the different algorithms which have been developed in the decades since the original inception of the fundamental power flow problem. Starting off with a concise formulation of the power flow problem, the solution algorithms are presented in a consistent step-by-step manner which aims to illustrate the similarities and differences between the different algorithms. An overview and discussion of results from different numerical methods is presented in section 4. Available data suggests that various Quasi-Newton methods come with drawbacks in convergence behaviour and do not necessarily provide the expected time savings even when they do converge. For reasons laid out in section 4.1, it proves very difficult to draw generalizable conclusions from multiple, independent works of research.

Keywords: Power system analysis, power flow calculation, Newton-Raphson, Gauss-Seidel, Holomorphic embedding

1 Introduction

Power flow calculation in symmetrical 3-phase AC systems is one of the most fundamental tools in the analysis of electrical power systems. It is essential to almost every step of power system planning and operation, starting with year-ahead power system expansion planning, all the way up to short-term operation planning and real-time power system operation. It forms the basis of optimal power flow and congestion management, and also provides a starting-off point for dynamic time-domain simulations of power systems by determining the initial system state. It is thus imperative that power flow calculation methods be as fast and as robust as possible.

1.1 Evaluation criteria for power flow calculation methods

1.1.1 Computing time

The criterion of speed is perhaps the easiest to evaluate as it is a tangible quantity which is (at least in principle) easy to measure and straight-forward to grasp conceptually. Consequently, most publications, including the ones covered by this review, focus on computing time when evaluating and comparing power flow algorithms.

1.1.2 Convergence properties

Perhaps harder to grasp and to quantify is the criterion of convergence behaviour. The most straight-forward case is non-convergence: When an iterative method does not converge to a solution of the power flow equations. If convergence is reached, one appropriate way to quantify convergence behaviour is to evaluate the change in power mismatch 3.1.2 across iterations until convergence. Alternatively, some publications simply list the number of iterations until convergence [Sau17, Mil09], Jan15]. This, however, is hardly relevant without specific information on how time-consuming a single iteration is: A large number of very quick iterations can be faster overall than a small number of very time-consuming iterations – in that case, the study of convergence behaviour has turned into an examination of computing time.

1.1.3 Solution accuracy (mismatch)

The third major evaluation criterion is solution accuracy, i.e. the remaining solution error (mismatch). This, again, is closely related to computing time: For iterative methods, a solution error threshold (typically in the range of 10^{-9} to 10^{-6} p.u.) usually is the primary convergence criterion and is thus a piece of input data to the algorithm, as it can be chosen by the user. A higher desired solution accuracy usually requires additional iterations (or other forms of computation) and thus additional computing time.

Solution accuracy becomes especially important when comparing iterative methods (for which solution accuracy can be chosen as a convergence threshold) to non-iterative methods such as linearized power flow approximations which inherently show a certain mismatch.

1.2 Notation conventions

Bold, italic denotes a vector ($\underline{\boldsymbol{U}}$); bold, upright denotes a matrix ($\underline{\mathbf{Y}}$). Underscores indicate complex variables, superscript * means complex conjugate.

2 The fundamental power flow problem

The purpose of this section is to provide a concise formulation of the fundamental problem of power flow and to introduce adequate notation and variables in order to allow for a sufficiently precise description of each algorithm in the following section. The basic set of power flow equations can be noted as

$$\underline{\boldsymbol{S}} = \mathrm{diag}\left(\underline{\boldsymbol{U}}\right) \cdot \underline{\boldsymbol{I}}^{*} = \mathrm{diag}\left(\underline{\boldsymbol{U}}\right) \cdot \underline{\mathbf{Y}}^{*} \cdot \underline{\boldsymbol{U}}^{*} \tag{1}$$

where $\underline{\boldsymbol{U}}$ is a vector containing the complex voltage at each bus in the grid and $\underline{\boldsymbol{S}}$ is a vector containing the complex apparent power injected into the grid at each bus. The matrix $\underline{\mathbf{Y}} = (\underline{y}_{ij})$ is the admittance matrix of the grid. Its elements are calculated by

$$\underline{y}_{ii} = \sum_{j} \underline{Y}_{ij}, \quad \underline{y}_{ii} = -\underline{Y}_{ij} \text{ for } i \neq j \tag{2}$$

where $\underline{Y}_{ij}$ is the admittance between buses i and j (for $i \neq j$) and $\underline{Y}_{ii}$ is the bus-to-ground shunt admittance at bus i.

This equation can be understood as requiring power balance at each bus (or more generally: energy conservation). It is the fundamental set of equations which must be satisfied in order to *solve* the power flow problem.

2.1 Bus-wise notation and separation into real and imaginary parts

Equation (1) denotes the power balance for all buses in the system. Let us now consider a single bus i (row i of the power balance equation in matrix form).

$$\underline{S_i} = \underline{U_i} \cdot \sum_j \underline{y_{ij}}^* \, \underline{U_j}^* \tag{3}$$

can be divided into its real and imaginary parts, active power P_i and reactive power Q_i

$$\underline{S_i} = P_i + \mathrm{j}Q_i \tag{4}$$

with

$$P_i = \sum_j U_i U_j g_{ij} \cos\left(\theta_i - \theta_j\right) + U_i U_j b_{ij} \sin\left(\theta_i - \theta_j\right) \tag{5}$$

$$Q_i = \sum_j U_i U_j g_{ij} \sin\left(\theta_i - \theta_j\right) - U_i U_j b_{ij} \cos\left(\theta_i - \theta_j\right) \tag{6}$$

where $\underline{U_i} = U_i \cdot \mathrm{e}^{\mathrm{j}\theta_i}$, $\underline{U_j} = U_j \cdot \mathrm{e}^{\mathrm{j}\theta_j}$ and $\underline{y_{ij}} = g_{ij} + \mathrm{j}b_{ij}$.

Using this notation, one can identify the four fundamental quantities for each bus i:

- The real power injected into the grid: P_i
- The reactive power injected into the grid: Q_i
- The voltage magnitude: U_i
- The voltage angle: θ_i

Given that there are four fundamental quantities but only two independent equations (5) and (6) for each bus, two out of four quantities must be known for each bus in order to have an equal number of knowns and unknowns. This leads to the common definition of bus types.

2.1.1 Bus types

The most commonly used bus types are PQ-, PV- and Slack buses, known and unknown quantities are listed in table 1.

Table 1: Bus types

Bus type	Known quantities	Unknown quantities
PQ	P_i, Q_i	U_i, θ_i
PV	U_i, P_i	Q_i, θ_i
Slack	U_i, θ_i	P_i , Q_i

Note: While some grid assets, such as synchronous generators with voltage control or constant power feeders fit rather well into the above definition of bus types, a significant amount of grid

assets cannot be easily and/or accurately modelled within this framework (e. g. transformers with on-load tap changers or modern FACTS devices such as STATCOMs). This will be disregarded for the remainder of this work. Furthermore, other real-world restrictions such as reactive power limits of generators will not be considered. Nevertheless, it is noteworthy that the inclusion of grid asset constraints and control mechanisms can have significant influence on power flow algorithm performance.

With the bus type definitions and the problem formulation now in place, the following section will introduce different solution algorithms.

3 Solution algorithms

Some of the equations for the algorithms in this section are (at least partly) presented in per-bus form. It should be duly noted that the most efficient implementations make use of vector-vector or vector-matrix operations, should avoid explicit loops and (if possible) if-statements [Kat22].

3.1 Iterative methods: Overview

All iterative methods follow the same basic scheme depicted in Figure 1.

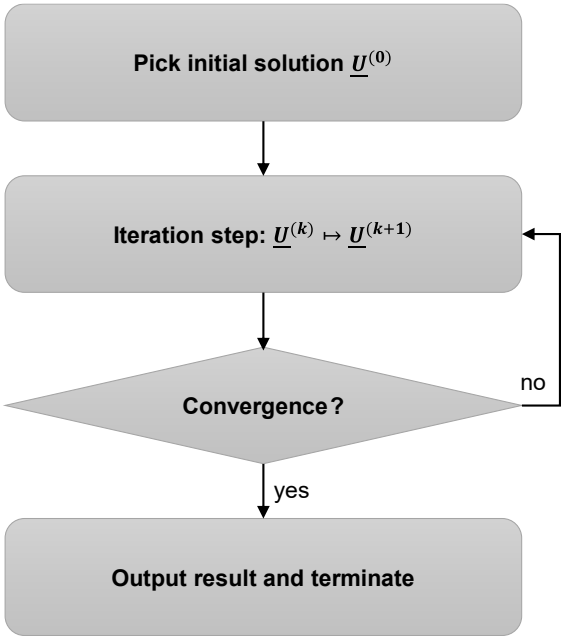

Figure 1: Basic iterative power flow algorithm scheme.

In practical implementations, there are of course more intricate details to the iterative process, such as iteration limiters. Those have been left out here for simplicity's sake and to highlight the three major parts of each iterative solution algorithm:

1. The initial solution
2. The solution update method (iteration step): $\underline{\boldsymbol{U}}^{(k)} \mapsto \underline{\boldsymbol{U}}^{(k+1)}$
3. The convergence criterion

The initial solution and convergence criterion will be discussed briefly in the following subsections. The solution update method is where iterative methods differ greatly, it is the subject of sections 3.2 to 3.4.

3.1.1 Initial solution

If no other information is available, the most common practice for “guessing” an initial solution is the so-called flat start, which sets $U_i = 1\,\mathrm{p.u.}$ and $\theta_i = 0$. Another widely used approach is to use the (already known / previously solved) solution $\underline{\widetilde{\boldsymbol{U}}}$ to a similar load or generation scenario $\underline{\tilde{\boldsymbol{S}}} = \mathrm{diag}\left(\underline{\widetilde{\boldsymbol{U}}}\right) \cdot \underline{\mathbf{Y}}^* \cdot \underline{\widetilde{\boldsymbol{U}}}^*$ as the initial solution for the given power flow problem, for example when analysing multiple consecutive load and generation scenarios [Dig24].

3.1.2 Convergence criterion

While other convergence criteria are possible, the one most commonly used [Sau17, Kat22] is power mismatch (for a Network with N buses):

$$\underline{S}_i - \sum_{j=1}^{N} \underline{U}_j \underline{y}_{ij} < \varepsilon \quad \forall i \in \{1, \dots, N\} \tag{7}$$

3.1.3 Iterative methods: Solution update method

The chosen iteration method has major influence on the algorithm’s computing time and convergence behaviour. They can fundamentally be categorized into fixed-point iteration methods, such as the Gauss-Seidel algorithm [Mil10, Sch23] (either using implementations based on the admittance matrix or the impedance matrix [Kat22, Sto74a]), and differential methods, i. e. the Newton-Raphson algorithm and other quasi-Newton methods [Tin67, Sch23, Sto74b, Sha68, Jan15, Bro65, Sau17]. Fixed-point methods typically lend themselves to easy implementation [Sto74a] and very quick iterations. However, due to their poorer (linear) convergence behaviour compared to differential methods (quadratic or super-linear convergence) [Sch23, Kat22], they require significantly more iterations until convergence. Moreover, considerations regarding implementation effort have become less relevant with the emergence of higher programming languages and standard libraries. Nevertheless, as shown in [Kat22], the specifics of implementation can have significant impact on algorithm performance.

3.2 Fixed point methods

It should be noted that nomenclature regarding the fixed point methods, especially the one presented in the following subsection 3.2.1, is inconsistent throughout the literature. While [Sau17] and others refer to the method presented in 3.2.1 as the Gauss-Seidel method, [Kat22] explicitly points out that the method is called the (admittance matrix) Jacobi method and that the Gauss-Seidel method actually refers to an algorithmic refinement of said Jacobi method.

3.2.1 Admittance matrix fixed-point algorithm

For the admittance matrix fixed-point algorithm, equation (3) is rearranged into a fixed point equation by considering its complex conjugate:

$$\underline{S_i}^* = P_i - \mathrm{j}Q_i = \underline{U_i}^* \cdot \sum_j y_{ij}\underline{U_j} = \underline{U_i}^* \cdot \left(\underline{y_{ii}}\underline{U_i} + \sum_{j \neq i} \underline{y_{ij}}\underline{U_j} \right) \tag{8}$$

And isolating $\underline{U_j}$ from the right side of the equation:

$$\underline{U_j} = \left(\frac{P_i - \mathrm{j}Q_i}{\underline{U_j}^*} - \sum_{j \neq i} \underline{y_{ij}}\underline{U_j} \right) \cdot \frac{1}{\underline{y_{ii}}} \tag{9}$$

The above equation is a fixed point equation $\underline{U_j} = f(\underline{U_j})$ and can be used for a fixed point iteration:

$$\underline{U_i}^{(k+1)} = \frac{1}{\underline{y_{ii}}} \left(\frac{P_i - \mathrm{j}Q_i}{\underline{U_i}^{(k)*}} - \sum_{j \neq i} \underline{y_{ij}}\underline{U_j}^{(k)} \right) \tag{10}$$

This iteration step is performed until convergence is reached (i. e. until equation (7) is satisfied). PV-buses (where Q_i is not specified) are handled by first estimating their reactive power infeed using equation (6) and the current voltage estimate $\underline{\boldsymbol{U}}^{(k)}$ and then computing the updated voltage angle for the PV-bus using equation (10) and disregarding the updated voltage magnitude (voltage magnitude is known for PV buses). [Sto74a, Sau17, Kat22]

3.2.2 Impedance matrix fixed point algorithm

Using the inverse of the admittance matrix $\underline{\mathbf{Y}}$, the impedance matrix $\underline{\mathbf{Z}} = \underline{\mathbf{Y}}^{-1} = (\underline{z}_{ij})$, an alternative fixed point algorithm can be formed [Sto74a, Kat22, Bro63]. Using $\underline{\boldsymbol{U}} = \underline{\mathbf{Z}} \cdot \underline{\boldsymbol{I}}$ or, in bus-wise notation, $\underline{U_i} = \sum_j \underline{z_{ij}}\underline{I_j}$, with $\underline{I_j} = \frac{P_j - \mathrm{j}Q_j}{\underline{U_j}^*}$, one obtains the following fixed point iteration:

$$\underline{U_i}^{(k+1)} = \sum_j \underline{z_{ij}} \frac{P_j - \mathrm{j}Q_j}{\underline{U_j}^{(k)*}} \tag{11}$$

[Kat22] points out that $\underline{\mathbf{Y}}$ is not generally invertible, and thus requires modification before inversion (two possible modifications are laid out in [Kat22]).

3.2.3 Updated impedance power flow

[Ou24] proposes an alternative to the standard admittance matrix fixed point method. Contrary to what this method's name might suggest, it does not use the networks impedance matrix (as described in 3.2.2). Constant power loads/injections are modelled as iteratively updated impedances. Given a specified nodal power $\underline{S_i} = P_i + \mathrm{j}Q_i$, its equivalent impedance in iteration k is

$$\underline{Z_i}^{(k)} = \frac{\left(\underline{U_i}^{(k)}\right)^2}{P_i - \mathrm{j}Q_i} \tag{12}$$

The equivalent impedances can now be considered regular shunt elements and can be included in the augmented system admittance matrix $\underline{\mathbf{Y_S}}$. It consists of the regular, unchanging admittance matrix $\underline{\mathbf{Y}}$ and the load admittance matrix

$$\underline{\mathbf{Y}}_{\mathbf{L}}{}^{(k)} = \mathrm{diag}\left(\frac{1}{\underline{Z}_i^{(k)}}\right) \tag{13}$$

$$\underline{\mathbf{Y}}_{\mathbf{S}}{}^{(k)} = \underline{\mathbf{Y}} + \underline{\mathbf{Y}}_{\mathbf{L}}{}^{(k)} \tag{14}$$

The elements of the augmented system admittance matrix $\underline{y}_{S,ij}$ can then be used in the reduced fixed point iteration

$$\underline{U}_i^{(k+1)} = -\frac{1}{\underline{y}_{\mathrm{S},ii}}\left(\sum_{j\neq i}\underline{y}_{\mathrm{S},ij}\underline{U}_j^{(k)}\right) \tag{15}$$

The updated impedance power flow and the impedance matrix fixed point algorithm can show improved convergence properties compared to the admittance matrix fixed point algorithm [Kat22, Sto74a] and the Newton-Raphson method [Ou24].

3.3 Newton-Raphson and quasi-Newton methods

3.3.1 Newton-Raphson algorithm

The general iteration rule of the Newton-Raphson algorithm [Kat22] can be stated as follows: For finding the roots of a function $f(x) = 0$, with a given iteration starting point $x^{(0)}$:

$$x^{(k+1)} = x^{(k)} - \left(\frac{\partial f}{\partial x}\left(x = x^{(k)}\right)\right)^{-1} \cdot f\left(x^{(k)}\right) \tag{16}$$

For functions of multiple variables, the derivative in the above equation becomes the Jacobian matrix $\mathbf{J}$.

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \left(\mathbf{J}\left(\boldsymbol{x} = \boldsymbol{x}^{(k)}\right)\right)^{-1} \cdot \boldsymbol{f}\left(\boldsymbol{x}^{(k)}\right) \tag{17}$$

For the power flow problem, equations (5) and (6) are rearranged into a root-finding problem, meaning the roots of the function $\boldsymbol{f}(\boldsymbol{U}, \boldsymbol{\theta})$ must be found:

$$\boldsymbol{f}(\boldsymbol{U}, \boldsymbol{\theta}) = \begin{pmatrix} \vdots \\ \sum_j U_i U_j g_{ij}\cos\left(\theta_i - \theta_j\right) + U_i U_j b_{ij}\sin\left(\theta_i - \theta_j\right) - P_i \\ \sum_j U_i U_j g_{ij}\sin\left(\theta_i - \theta_j\right) - U_i U_j b_{ij}\cos\left(\theta_i - \theta_j\right) - Q_i \\ \vdots \end{pmatrix} = \mathbf{0} \tag{18}$$

The iteration step in equation (17) can now be applied with $\boldsymbol{x} = \begin{pmatrix}\boldsymbol{U}\\ \boldsymbol{\theta}\end{pmatrix}$. While the bus-wise notation might be suitable for understanding the process in principle, [Kat22] points out that an efficient implementation on modern hardware should employ vector-vector or vector-matrix operations and make use of modern computers' capabilities of handling complex numbers directly. This might require a reordering of the components of $\boldsymbol{f}$ and $\boldsymbol{x}$, but the algorithm does not change in principle.

PV buses are typically handled by dropping the reactive power balance equation, thus reducing the equation system from $2(N-1)$ equations and $2(N-1)$ unknowns to $2N_{\mathrm{PQ}}+N_{\mathrm{PV}}$ equations and unknowns (for a system with N buses, N_{PQ} PQ buses and N_{PV} PV buses, $N=N_{\mathrm{PQ}}+N_{\mathrm{PV}}+1$, the last bus being the slack bus).

3.3.2 Newton-Raphson algorithm with constant matrices ("dishonest" Newton-Raphson / chord method)

Since the re-evaluation of the Jacobian and the solution of the linear system of equations in (17) constitute a major part of the computational burden of the Newton-Raphson algorithm, there are several approaches which aim to reduce the amount of times that the Jacobian must be updated. One such approach consists of calculating and factorizing the Jacobian only once, in the first iteration, and reusing it until convergence [Sau17, San20, Jan15].

3.3.3 Newton-Raphson algorithm with statically updated Jacobian matrix (Shamanskii's method)

A less radical approach is Shamanskii's method [Sha68] in which the Jacobian is updated after a fixed (specified) number of iterations.

3.3.4 Newton-Raphson method with dynamically updated Jacobian matrix

The *Combined Newton-Raphson Method* [Jan15] updates the Jacobian dynamically if the rate of convergence is too low. The condition for updating the Jacobian is as follows:

$$\frac{\max\left(\boldsymbol{f}\left(\boldsymbol{x}^{(k-2)}\right)\right)}{\max\left(\boldsymbol{f}\left(\boldsymbol{x}^{(k)}\right)\right)} \leq \lambda \tag{19}$$

Where $\lambda > 1$ is the specified minimum convergence rate within the last two iterations (e.g. $\lambda = 1.2$).

3.3.5 Broyden's method

Broyden's method [Bro65, San20, Osa05] proposes an iterative approximation for directly computing the inverse of the Jacobian $\left(\mathbf{J}^{(k)}\right)^{-1} = \left(\mathbf{J}(\boldsymbol{x}^{(k)})\right)^{-1}$ from the inverse of the Jacobian of the previous iteration, thus eliminating the need to explicitly re-evaluate and factorize the Jacobian in every iteration and reducing the solution of the linear system in (17) to a simple matrix multiplication. Broyden proposes two alternative ways of calculating $\left(\mathbf{J}^{(k)}\right)^{-1}$:

$$\left(\mathbf{J}^{(k)}\right)^{-1} = \left(\mathbf{J}^{(k-1)}\right)^{-1} + \frac{\Delta\boldsymbol{x}^{(k)} - \left(\mathbf{J}^{(k-1)}\right)^{-1}\Delta\boldsymbol{f}^{(k)}}{\Delta\boldsymbol{x}^{(k)^{\mathrm{T}}}(\mathbf{J}^{(k-1)})^{-1}\Delta\boldsymbol{f}^{(k)}} \cdot \Delta\boldsymbol{x}^{(k)^{\mathrm{T}}}\left(\mathbf{J}^{(k-1)}\right)^{-1} \tag{20}$$

or

$$\left(\mathbf{J}^{(k)}\right)^{-1} = \left(\mathbf{J}^{(k-1)}\right)^{-1} + \frac{\Delta\boldsymbol{x}^{(k)} - \left(\mathbf{J}^{(k-1)}\right)^{-1}\Delta f^{(k)}}{\left\|\Delta\boldsymbol{f}^{(k)}\right\|^{2}} \cdot \Delta\boldsymbol{f}^{(k)^{\mathrm{T}}} \tag{21}$$

Where $\Delta\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}$ and $\Delta\boldsymbol{f}^{(k)} = \boldsymbol{f}\left(\boldsymbol{x}^{(k)}\right) - \boldsymbol{f}\left(\boldsymbol{x}^{(k-1)}\right)$.

3.3.6 Decoupled Newton-Raphson algorithm with constant matrices ("fast decoupled load flow")

The Jacobian in (17) contains the partial derivatives of the scalar components f_i of $\boldsymbol{f}$ (nodal active and reactive power balances ΔP_i and ΔQ_i) with respect to voltage magnitudes U_i and angles θ_i, i.e. the partial derivatives of the active and reactive power balance equations (5) and (6): $\frac{\partial P_i}{\partial \theta_j}$, $\frac{\partial P_i}{\partial U_j}$, $\frac{\partial Q_i}{\partial \theta_j}$, and $\frac{\partial Q_i}{\partial U_j}$. As the coupling between P and U and between Q and θ can be considered weak, the respective partial derivatives can be neglected: $\frac{\partial P_i}{\partial U_j} = \frac{\partial Q_i}{\partial \theta_j} = 0$. Furthermore, assuming small voltage angle differences between neighbouring buses $\theta_i \approx \theta_j$ and thus $\sin(\theta_i - \theta_j) \approx 0$ and $\cos(\theta_i - \theta_j) \approx 1$, (17) dissolves into two separate linear systems for iterative voltage angle and voltage magnitude updates:

$$\Delta\boldsymbol{\theta}^{(k)} = -(\mathbf{B}')^{-1} \cdot \widetilde{\mathbf{U}}^{(\boldsymbol{k})} \cdot \Delta\boldsymbol{P}^{(k)} \tag{22}$$

$$\Delta\boldsymbol{U}^{(k)} = -(\mathbf{B}'')^{-1} \cdot \widetilde{\mathbf{U}}^{(\boldsymbol{k})} \cdot \Delta\boldsymbol{Q}^{(k)} \tag{23}$$

with $\widetilde{\mathbf{U}}^{(\boldsymbol{k})} = \mathrm{diag}\left(\frac{1}{U_1^{(k)}}, \ldots, \frac{1}{U_N^{(k)}}\right)$ and $(\mathbf{B}')_{ii} = \sum_j b_{ij}$, $(\mathbf{B}')_{ij} = b_{ij}$ for $i \neq j$, $(\mathbf{B}'')_{ij} = -b_{ij}$. The elements of the matrices $\mathbf{B}'$ and $\mathbf{B}''$ can be computed directly from the imaginary parts of the admittance matrix elements y_{ij}. Since the matrices are constant, they only need to be computed and factorized once, reducing computing time per iteration. [Sch23, Sto74b]

3.3.7 Continuous Newton method

When considering a set of ordinary differential equations

$$\dot{\boldsymbol{x}} = \boldsymbol{g}(\boldsymbol{x}) \tag{24}$$

the explicit Euler method for numerical integration states (for a given time step Δt):

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \Delta t \boldsymbol{g}\left(\boldsymbol{x}^{(k)}\right) \tag{25}$$

When using $\Delta t = 1$ and defining

$$\boldsymbol{g}(\boldsymbol{x}) = -\left(\mathbf{J}\left(\boldsymbol{x}^{(k)}\right)\right)^{-1} \cdot \boldsymbol{f}\left(\boldsymbol{x}^{(k)}\right) \tag{26}$$

equation (17) can be interpreted as the k-th step of the explicit Euler method. The equilibrium point $\boldsymbol{x_0}$ of (24) where

$$\boldsymbol{g}(\boldsymbol{x_0}) = \mathbf{0} \tag{27}$$

is also a solution to the root-finding problem (18) and thus a solution to the power flow problem. This analogy suggests that numerical methods can be used to integrate (24) to find the equilibrium point. [Mil09], e.g., uses a fourth order Runge-Kutta formula.

3.3.8 Improving convergence radius of the classical Newton-Raphson method

In some cases where the classical Newton-Raphson algorithm 3.3.1 fails to converge, even though a solution exists, convergence can be improved by limiting the step size

$$x^{(k+1)} = x^{(k)} - \mu\left(\frac{\partial f}{\partial x}\left(x = x^{(k)}\right)\right)^{-1} \cdot f\left(x^{(k)}\right) \tag{28}$$

with $\mu < 1$. There are several techniques to determine adequate values for μ, some of which include optimization and are thus called optimal multiplier methods [Bij03, Iwa78, Iwa81, Tat05, Tyl92]. It can be advantageous to start the iterative process with a limited step size ($\mu < 1$) and then, after some iterations, return to the standard Newton-Raphson process ($\mu = 1$).

3.4 The holomorphic embedding load flow method

The holomorphic embedding load flow method [Tri12] introduces a complex parameter $\underline{s}$ such that the bus voltages become holomorphic functions of $\underline{s}$ and evaluating $\underline{U}_i(\underline{s} = 1)$ yields the original power flow problem (3). Equation (3) can be rearranged into

$$\frac{\underline{S}_i^*}{\underline{U}_i^*} = \sum_j \underline{y}_{ij}\underline{U}_j \tag{29}$$

The embedding proposed by [Tri12] is

$$\frac{\underline{s}\underline{S}_i^*}{\underline{U}_i^*(\underline{s}^*)} = \sum_j \underline{y}_{ij}\underline{U}_j(\underline{s}) \tag{30}$$

Further reasoning why this embedding makes $\underline{U}_i(\underline{s})$ and $\underline{U}_i^*(\underline{s}^*)$ holomorphic functions can be found in [Tri12]. As holomorphic functions, $\underline{U}_j(\underline{s})$ and $\frac{1}{\underline{U}_i(\underline{s})}$ can be expressed as complex power series:

$$\underline{U}_j(\underline{s}) = \sum_{n=0}^{\infty} \underline{c}_j[n]\underline{s}^n \tag{31}$$

$$\frac{1}{\underline{U}_j(\underline{s})} = \sum_{n=0}^{\infty} \underline{d}_i[n]\underline{s}^n \tag{32}$$

Inserting these power series into (30) yields:

$$\underline{s}\underline{S}_i^* \sum_{n=0}^{\infty} \underline{d}_i^*[n]\underline{s}^n = \sum_j \underline{y}_{ij} \sum_{n=0}^{\infty} \underline{c}_j[n]\underline{s}^n \tag{33}$$

Taking the derivative of (33) with respect to $\underline{s}$ and setting $\underline{s} = 0$, one obtains:

$$\sum_j \underline{y}_{ij}\, \underline{c}_j[n] = \underline{S}_i^* \underline{d}_i^*[n-1] \tag{34}$$

And finally, by using

$$1 = \underline{U}(\underline{s})\frac{1}{\underline{U}(\underline{s})} = \left(\sum_{n=0}^{\infty} \underline{c}[n]\underline{s}^n\right)\left(\sum_{n=0}^{\infty} \underline{c}[n]\underline{s}^n\right)$$

$$= \underline{c}_0\underline{d}_0 + \underline{s}\sum_{m=0}^{1} \underline{c}[1-m]\underline{d}[m] + \underline{s}^2 \sum_{m=0}^{2} \underline{c}[2-m]\underline{d}[m] + \cdots + \sum_{m=0}^{n} \underline{c}[n-m]\underline{d}[m] + \ldots \tag{35}$$

and comparing coefficients on each side, one obtains:

$$\underline{d}_i[n] = -\frac{\sum_{m=0}^{n-1} \underline{c}_i[n-m]\underline{d}_i[m]}{\underline{c}_i[0]} \tag{36}$$

Equations (34) and (36) can be used to compute coefficients $\underline{c}[n]$ and of any desired order. Coefficients $\underline{c}_i[0]$ are obtained by setting $\underline{s} = 0$ in equation (33) and solving $\sum_j \underline{y}_{ij}\underline{c}_i = 0$. The power series (31) is not computed directly, but by means of Padé-Approximants. While this algorithm is generally understood to be non-iterative for a given order of the power series n, it can be viewed as an iterative procedure in the following sense: The power series coefficients are computed up to order k, the solution for this order $\underline{\boldsymbol{U}}^{(k)}$ is then computed and evaluated for mismatch (equation (7)). If the mismatch is too large, coefficients for order $k+1$ are computed, and the evaluation is repeated. This fits neatly into the iterative scheme in Figure 1, with the exceptions that no initial solution must be chosen and $\underline{\boldsymbol{U}}^{(k+1)}$ is independent of $\underline{\boldsymbol{U}}^{(k)}$**.**

4 Performance and robustness evaluation from available literature

This section provides a summary of the findings from the various works examined in this paper.

4.1 General remarks

In a sense, one of the key insights of this work is that a fair, accurate, detailed and generalizable comparison of power flow algorithms based on the available literature proves notoriously difficult for several reasons, most of which relate to inconsistencies between different (numerical) studies:

1. Inconsistent investigation frameworks/test systems. Any numerical study of power flow algorithms requires at least one (or ideally many) grid models and corresponding load and generation scenarios. While many works rely on a somewhat standardized and publicly accessible set of test systems, such as the IEEE test systems or test systems from open source libraries [Zim11, Zim25, Bro18, Jos16], not all studies are performed on unmodified versions of these test systems. Available data also suggests that superficial criteria such as the number of buses of a system is often insufficient to allow for comparisons, meaning that, e. g., two systems with a similar number of buses can require significantly different computation effort.
2. Inconsistent evaluation criteria. Evaluation criteria are introduced in section 1.1. While computing time can be a valid point of comparison within a single numerical study, its value diminishes when trying to compare algorithms across different works of literature. This is mostly due to the noticeable influence which the chosen software, hardware and the specifics of the implementation have on an algorithm's computation time [Kat22], [Sch18].

Moreover, even when testing on multiple systems, numerical studies often rely on a very limited number of generation and load scenarios (in most cases: just one per test system) which calls generalized conclusions from these studies into question.

4.2 Standard algorithm comparison

[Sau17] provides a comparison between several standard power flow algorithms on a variety of small and large test cases, showing that:

- For small grids (around or fewer than 100 buses), the admittance matrix fixed-point algorithm wins out over Newton-Raphson with regard to performance, requiring only 60 % of computing time compared to standard Newton-Raphson.
- For large grids (2000 to 3000 buses), the reverse is true, with Newton-Raphson requiring about 43 % of computing time compared to the admittance matrix fixed point algorithm. Although, for 2 out of 7 large test grids, Newton-Raphson does not converge (where the fixed-point method does).
- Both of the above findings are supported by the analysis performed in [Kat22].
- The holomorphic embedding method, even when calculating coefficients $\underline{c}[n]$ up to an order of $n = 20$, does not reach the same average solution accuracy as iterative methods, implying that the holomorphic embedding method can achieve either the same speed or the same solution accuracy as standard iterative methods – but perhaps not both.
- The fast decoupled load flow fails to converge for 6 out of 7 large test grids and requires the highest computing time when it does converge, providing strong evidence that the improvement in computing time per iteration does not make up for the worsened convergence properties.

[Kat22] compares the Newton-Raphson algorithm, the admittance and the impedance matrix fixed point algorithms on a large distribution grid model (around 900 buses) and finds that the impedance matrix fixed point algorithm possesses a significant advantage over the Newton-Raphson algorithm (around 150 times faster), although it is explicitly stated that the investigated test grid plays to the strengths of this method (no PV buses, no change in network admittance, typical distribution grid topology). It remains to be seen whether these results can be extrapolated to highly meshed, high voltage transmission grids with a large(r) number of PV-buses.

4.3 Effects of reducing Jacobian updates in the Newton-Raphson algorithm

The methods listed in sections 3.3.2, 3.3.3, and 3.3.4 all aim to reduce computing time by limiting the amount of times the Jacobian matrix is updated.

[Jan15] shows that the „dishonest“ Newton-Raphson method which calculates and factorizes the Jacobian only once has trouble converging at all, especially for larger grids of several thousand buses. These findings are supported by [Sau17]. Shamanskii’s method achieves a significant speed-up over Newton-Raphson especially for the larger test cases investigated in [Jan15], while the best overall results are (somewhat expectedly) achieved with dynamically updating the Jacobian as it adapts to the problem at hand (see section 3.3.4).

4.4 Broyden’s method

Broyden’s method is investigated in a very limited manner in [San20] and [Osa05]. While the presentation in both papers is sound, the investigative framework is limited to one small test

grid in [Osa05] and 4 small test grids in [San20]. Results from both references indicate that Broyden's method performs worse than the classical Newton-Raphson method. Although, as discussed in the previous two sections, observations regarding the relative performance and convergence of algorithms can vary depending on grid size. The authors of this paper view the results presented in [San20] and [Osa05] as non-conclusive.

4.5 Continuous Newton method

[Mil09] shows that the continuous Newton method outperforms other methods which aim to improve power flow convergence. However, due to the limitation to only one (realistically large) test system where the standard Newton-Raphson method does not converge, no direct comparison is possible. Given that the fourth order Runge-Kutta method proposed in [Mil09] requires four factorizations of the Jacobian per iteration (whereas the standard Newton-Raphson method requires just one per iteration), only a higher convergence rate would make it computationally competitive to the standard Newton-Raphson method.

5 Concluding remarks

The purpose of this paper was to provide an overview of power flow calculation methods and present each method using a consistent notation across all of them. The authors' secondary intention was to analyse the numerical results available in the literature. As was pointed out in section 4.1, it proves notoriously difficult to draw generalizable conclusions across multiple works of literature. Given the extreme care needed and the amount of influencing factors to consider when trying to accurately benchmark power flow algorithms (or any other algorithms) on modern computers, as is laid out in detail by [Kat22], a generalizable comparison between all algorithms covered in this paper alone probably exceeds the extent of a single research contribution.

6 References

[Bij03] Bijwe, P. R.; Kelapure, S. M.: Nondivergent fast power flow methods. In IEEE Transactions on Power Systems, 2003, 18; pp. 633–638.

[Bro18] Brown, T.; Hörsch, J.; Schlachtberger, D.: PyPSA: Python for Power System Analysis. In Journal of Open Research Software, 2018, 6; p. 4.

[Bro63] Brown, H.; Carter, G.; Happ, H.; Person, C.: Power Flow Solution by Impedance Matrix Iterative Method. In IEEE Transactions on Power Apparatus and Systems, 1963, 82; p. 1–10.

[Bro65] Broyden, C. G.: A Class of Methods for Solving Nonlinear Simultaneous Equations. In Mathematics of Computation, 1965, 19; p. 577.

[Dig24] PowerFactory 2024 User Manual, Gomaringen, 2024.

[Iwa78] Iwamoto, S.; Tamura, Y.: A Fast Load Flow Method Retaining Nonlinearity. In IEEE Transactions on Power Apparatus and Systems, 1978, PAS-97; pp. 1586–1599.

[Iwa81] Iwamoto, S.; Tamura, Y.: A Load Flow Calculation Method for Ill-Conditioned Power Systems. In IEEE Transactions on Power Apparatus and Systems, 1981, PAS-100; pp. 1736–1743.

[Jan15] Janković, S.; Ivanović, B.: Application of combined Newton–Raphson method to large load flow models. In Electric Power Systems Research, 2015, 127; pp. 134–140.

[Jos16] Josz, C.; Fliscounakis, S.; Maeght, J.; Panciatici, P.: AC Power Flow Data in MATPOWER and QCQP Format: iTesla, RTE Snapshots, and PEGASE. arXiv, 2016.

[Kat22] Kattmann, C.: Optimization of Power Flow Computation Methods. Dissertation, Stuttgart, 2022.

[Mil09] Milano, F.: Continuous Newton's Method for Power Flow Analysis. In IEEE Transactions on Power Systems, 2009, 24; pp. 50–57.

[Mil10] Milano, F.: Power system modelling and scripting. Springer, Heidelberg, 2010.

[Osa05] Osaloni, O.; Radman, G.: Integrated AC/DC systems power flow solution using newton-raphson and broyden approaches: Proceedings of the Thirty-Seventh Southeastern Symposium on System Theory, 2005. SSST '05. IEEE, 2005; pp. 225–229.

[Ou24] Ou, B.; Wang, B.; Cui, B.; Wu, D.: Updated Impedance Power Flow: 2024 56th North American Power Symposium (NAPS). IEEE, 2024; pp. 1–6.

[San20] Sangadiev, A.; Poddubny, A.; Pozo, D.; Gonzalez-Castellanos, A.: Quasi-Newton Methods for Power Flow Calculation: 2020 International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE). IEEE, 2020; pp. 1–6.

[Sau17] Sauter, P. S.; Braun, C. A.; Kluwe, M.; Hohmann, S.: Comparison of the Holomorphic Embedding Load Flow Method with Established Power Flow Algorithms and a New Hybrid Approach: 2017 Ninth Annual IEEE Green Technologies Conference (GreenTech). IEEE, 2017; pp. 203–210.

[Sch18] Schafer, F.; Braun, M.: An efficient open-source implementation to compute the jacobian matrix for the Newton-Raphson power flow algorithm: 2018 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe). IEEE, 2018; pp. 1–6.

[Sch23] Schäfer, K. F.: Netzberechnung. Verfahren zur Berechnung elektrischer Energieversorgungsnetze. Springer Vieweg, Wiesbaden, Heidelberg, 2023.

[Sha68] Shamanskii, V. E.: A modification of Newton's method. In Ukrainian Mathematical Journal, 1968, 19; pp. 118–122.

[Sto74a] Stott, B.: Review of load-flow calculation methods. In Proceedings of the IEEE, 1974a, 62; pp. 916–929.

[Sto74b] Stott, B.; Alsac, O.: Fast Decoupled Load Flow. In IEEE Transactions on Power Apparatus and Systems, 1974b, PAS-93; pp. 859–869.

[Tat05] Tate, J. E.; Overbye, T. J.: A Comparison of the Optimal Multiplier in Polar and Rectangular Coordinates. In IEEE Transactions on Power Systems, 2005, 20; pp. 1667–1674.

[Tin67] Tinney, W.; Hart, C.: Power Flow Solution by Newton's Method. In IEEE Transactions on Power Apparatus and Systems, 1967, PAS-86; pp. 1449–1460.

[Tri12] Trias, A.: The Holomorphic Embedding Load Flow method: 2012 IEEE Power and Energy Society General Meeting. San Diego, California, USA, 22-26 July 2012. IEEE, [Piscataway, N.J.], 2012; pp. 1–8.

[Tyl92] Tylavsky, D. J.; Crouch, P.; Jarriel, L. F.; Adapa, R.: Improved power flow robustness for personal computers. In IEEE Transactions on Industry Applications, 1992, 28; pp. 1102–1108.

[Zim11] Zimmerman, R. D.; Murillo-Sanchez, C. E.; Thomas, R. J.: MATPOWER: Steady-State Operations, Planning, and Analysis Tools for Power Systems Research and Education. In IEEE Transactions on Power Systems, 2011, 26; pp. 12–19.

[Zim25] Zimmerman, R. D.; Murillo-Sánchez, C. E.: MATPOWER. Zenodo, 2025.