Tensor power flow formulations for multidimensional analyses in distribution systems

Duque, Edgar Mauricio Salazar; Giraldo, Juan S.; Vergara, Pedro P.; Nguyen, Phuong H.; Slootweg, Han (J G.).

DOI
10.1016/j.ijepes.2024.110275

Publication date
2024

Document Version
Final published version

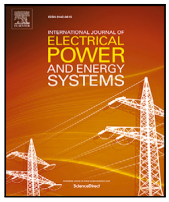
Published in
International Journal of Electrical Power and Energy Systems

Electrical Power and Energy Systems 162 (2024) 110275

Contents lists available at ScienceDirect

International Journal of Electrical Power and Energy Systems

journal homepage: www.elsevier.com/locate/ijepes



Tensor power flow formulations for multidimensional analyses in distribution systems

Edgar Mauricio Salazar Duque [a], Juan S. Giraldo [b], Pedro P. Vergara [c,*], Phuong H. Nguyen [a], Han (J.G.) Slootweg [a]

[a] Electrical Energy Systems Group, Eindhoven University of Technology, Eindhoven, 5612 AE, The Netherlands
[b] Energy Transition Studies, Netherlands Organization for Scientific Research (TNO), Amsterdam, 1043 NT, The Netherlands
[c] Department of Electrical Sustainable Energy, Delft University of Technology, Delft, 2628 CD, The Netherlands

ARTICLE INFO

Keywords:
Power flow
Fixed-point iteration
Tensor
Mixed computer resources
GPU

ABSTRACT

In this paper, we present two multidimensional power flow formulations based on a fixed-point iteration (FPI) algorithm to efficiently solve hundreds of thousands of Power flows (PFs) in distribution systems. The presented algorithms are the base for a new TensorPowerFlow (TPF) tool and shine for their simplicity, benefiting from multicore Central processing unit (CPU) and Graphics processing unit (GPU) parallelization. We also focus on the mathematical convergence properties of the algorithm, showing that its unique solution is at the practical operational point. The proof is validated using numerical simulations showing the robustness of the FPI algorithm compared to the classical Newton–Raphson (NR) approach. In the case study, a benchmark with different PF solution methods is performed, showing that for applications requiring a yearly simulation at 1-minute resolution, the computation time is decreased by a factor of 164, compared to the NR in its sparse formulation. Finally, a set of applications is described, highlighting the potential of the proposed formulations over a wide range of analyses in distribution systems.

1. Introduction

The power flow study is crucial for different technical analyses of electrical distribution systems. For example, power flows (PFs) are widely used in Time series simulation (TSS), where long-term analysis depends on high granularity in time, e.g., integration of distributed energy resources, Volt/Var control, and hosting capacity [1]. Another widespread use of multiple power flows is the Probabilistic power flow (PPF), in which exogenous uncertainties are modeled using scenarios, evaluated using power flows, and its impact evaluated using stochastic analysis [2]. Many other applications require multiple power flow executions, such as metaheuristic-based optimization, contingency analysis, and machine learning in power systems [3].

Notably, the common factor between the aforementioned applications is the execution of multiple (thousands or even millions) power flows to provide insightful results. These applications are *multidimensional* regardless of the technique used to reduce the number of timestamps, scenarios, or training size [1–3]. The multidimensionality motivates efficient power flow formulations and new techniques, providing fast and accurate results. Multidimensional problems have been tackled in the past, such as in [2] for the PPF problem using the embedded holomorphic power flow, or in [4] where tensors are used to formulate the three-phase power flow. As expressed by the authors in [4], using the tensor formulation reduced the memory requirement and improved the calculation times compared to the traditional implementation. However, the authors reported that computational times underperformed the ones with the BFS. On the other hand, authors in [2] found their algorithm to be more computationally exhaustive than the compared alternatives. These conclusions motivated the implementation of a computationally efficient tensor formulation for multidimensional problems.

Advances in computer hardware, such as the increase in the number of cores in CPUs and the evolution of GPU designs, transitioning from simple graphics processors to highly parallel multiprocessors of many cores [5], opened a new paradigm of programming and rethinking PF algorithms. New lines of research look at reducing computational time by combining CPU and GPU resources to improve the speed of convergence of Newton–Raphson algorithms [6–9], which is seen as a preferred method to improve due to its quadratic convergence. However, the formulation of these approaches is designed specifically to solve one power flow in the least amount of time. Unfortunately, these algorithms are heavily focused on GPU-based architectures, making

* Corresponding author.
E-mail address: P.P.VergaraBarrios@tudelft.nl (P.P. Vergara).

https://doi.org/10.1016/j.ijepes.2024.110275
Received 11 April 2024; Received in revised form 23 July 2024; Accepted 26 September 2024
Available online 19 October 2024

their implementation a tedious process and requiring specific GPU programming knowledge. However, applying general mathematical formulations of power flow to multidimensional problems using new hardware capabilities is relatively new.

Fixed-point iteration (FPI) algorithms have been proposed to solve the PF problem, showing accurate results and numerical stability. As described in [10,11], FPI methods can guarantee convergence to the power flow solution via the Banach fixed-point theorem. In this paper, we advocate using an FPI algorithm to solve multidimensional PF formulations. This fixed point algorithm has shown robust performance [11]; it has a simple formulation for the case of distribution system analyses,[1] it is suitable and scalable for multidimensional applications in the form of a tensor, and can also benefit from multicore CPU and GPU parallelization. In this work, we also focus on the mathematical convergence properties of the algorithm, showing its unique point of solution at the practical operational point (if the solution exists), which is the high-voltage, low-current solution. The contributions of the paper are as follows:

- Present a practical tool for multidimensional power flow analysis in distribution systems, named TensorPowerFlow (TPF)[2] in its dense and sparse versions, which are based on a FPI algorithm. The TPF opens new possibilities for using mixed computing resources (CPUs or GPUs) to increase performance.
- Give a geometric and physical interpretation of the existence of the power flow solution using an FPI iterative algorithm, showing the robustness and numerical stability of the FPI algorithm.

2. Single dimension fixed point power flow

We take a network with one substation, $b = |\Omega_{\mathrm{d}}|$ demand nodes, and $\phi = |\Omega_{\phi}|$ phases. Nodal voltages and currents are related by the admittance matrix $\boldsymbol{Y} \in \mathbb{C}^{(b+1)\phi\times(b+1)\phi}$ as follows:

$$\begin{bmatrix} \boldsymbol{i}_s \\ -\boldsymbol{i}_d \end{bmatrix} = \begin{bmatrix} \boldsymbol{Y}_{ss} & \boldsymbol{Y}_{sd} \\ \boldsymbol{Y}_{ds} & \boldsymbol{Y}_{dd} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_s \\ \boldsymbol{v}_d \end{bmatrix} \tag{1}$$

where vectors $\boldsymbol{i}_s \in \mathbb{C}^{\phi\times 1}$ and $\boldsymbol{i}_d \in \mathbb{C}^{b\phi\times 1}$ represent complex injections of nodal current at the substation and at the demand nodes $d \in \Omega_{\mathrm{d}}$, while $\boldsymbol{v}_s \in \mathbb{C}^{\phi\times 1}$ and $\boldsymbol{v} := \boldsymbol{v}_d \in \mathbb{C}^{b\phi\times 1}$ are complex components of the respective nodal voltages. It must be noted that (1) is general and can represent single-phase, polyphase, radial, meshed networks and distributed generation of constant power.

In this paper, we call a *single-dimensional* power flow to the solution of (1) for a single vector of nominal complex power $\boldsymbol{s} \in \mathbb{C}^{b\phi\times 1}$ at the demand nodes, which is a snapshot of consumption in the grid. The single-dimensional power flow formulation, namely the FPI algorithm, is the basis of the tensorized version, and it is based on the notation of the equivalent version of the successive approximation method (SAM) presented in [11] as:

$$\boldsymbol{v}_{(n+1)} = \boldsymbol{F}\boldsymbol{v}_{(n)}^{*\circ(-1)} + \boldsymbol{w} \tag{2}$$

with

$$\begin{aligned} &\boldsymbol{A} = \mathbf{diag}\left(\alpha_P \odot \boldsymbol{s}^*\right); \quad \boldsymbol{B} = \mathbf{diag}\left(\alpha_Z \odot \boldsymbol{s}^*\right) + \boldsymbol{Y}_{dd}; \\ &\boldsymbol{c} = \boldsymbol{Y}_{ds}\boldsymbol{v}_s + \alpha_I \odot \boldsymbol{s}^*; \\ &\boldsymbol{F} = -\boldsymbol{B}^{-1}\boldsymbol{A}; \qquad \boldsymbol{w} = -\boldsymbol{B}^{-1}\boldsymbol{c} \end{aligned} \tag{3}$$

where $\odot$ is the Hadamard product, $\boldsymbol{v}_{(n)}^{*\circ(-1)}$ the element-wise reciprocal of the conjugated vector containing the nodal voltages at iteration n, and α_Z, α_I, and α_P represent the coefficients of the ZIP load model per demand node and phase. It is worth mentioning that $\boldsymbol{A} \in \mathbb{C}^{b\phi\times b\phi}$, $\boldsymbol{B} \in \mathbb{C}^{b\phi\times b\phi}$, and $\boldsymbol{c} \in \mathbb{C}^{b\phi\times 1}$ are constant matrices and vector within the iterative process for a particular operating point. Thus, $\boldsymbol{F} \in \mathbb{C}^{b\phi\times b\phi}$ and $\boldsymbol{w} \in \mathbb{C}^{b\phi\times 1}$ can be calculated once and used at each iteration.

The algorithm in (2) can be rearranged as an iterative mapping using $\boldsymbol{v}_{(n)}^{*\circ-1} = \boldsymbol{v}_{(n)} \oslash \|\boldsymbol{v}_{(n)}\|_{\circ}^2$, where $\oslash$ is the Hadamard division and $\|\cdot\|_{\circ}$ is the element-wise euclidean norm, as

$$\begin{aligned} \boldsymbol{v}_{(n+1)} &= -\boldsymbol{B}^{-1}\,\mathbf{diag}(\alpha_p \odot \boldsymbol{s}^* \oslash \|\boldsymbol{v}_{(n)}\|_{\circ}^2)\boldsymbol{v}_{(n)} + \boldsymbol{w} \\ &= T(\boldsymbol{v}_{(n)}) \end{aligned} \tag{4}$$

Here, the Banach fixed-point theorem is used to prove that (4) is a contraction mapping in order to have a solution. This is the case where $T(\cdot)$ satisfies

$$\|\boldsymbol{v}_{(n+1)} - \boldsymbol{v}_{(n)}\|_1 \le k\|T(\boldsymbol{v}_{(n+1)}) - T(\boldsymbol{v}_{(n)})\|_1, \tag{5}$$

where the norm-1 distance is used in our case ($\|\cdot\|_1$), that is, for vectors $\|\boldsymbol{x}\|_1 = \sum_i |x_i|$, where $|\cdot|$ is the absolute value, and for matrices $\|A\|_1 = \max_j(\sum_i |a_{ij}|)$, which is the column norm. When $k < 1$, the contraction mapping $T(\cdot)$ has a unique point of convergence. This can be shown as

$$\begin{aligned} \|\boldsymbol{v}_{(n+1)} - \boldsymbol{v}_{(n)}\|_1 = \|&\boldsymbol{B}^{-1}\,\mathbf{diag}(\alpha_p \odot \boldsymbol{s}^* \oslash \|\boldsymbol{v}_{(n+1)}\|_{\circ}^2)\boldsymbol{v}_{(n+1)} \\ &- \boldsymbol{B}^{-1}\,\mathbf{diag}(\alpha_p \odot \boldsymbol{s}^* \oslash \|\boldsymbol{v}_{(n)}\|_{\circ}^2)\boldsymbol{v}_{(n)}\,\|_1 \end{aligned} \tag{6}$$

Assuming that $\boldsymbol{s}$ is a feasible load consumption (details of this feasibility are given in Section 3), the iterative algorithm reaches closer voltage values at each iteration, i.e., $\boldsymbol{v}_{(n+1)} \mapsto \boldsymbol{v}_{(n)}$. Which means that after a number of iterations, the voltage at the solution is $\boldsymbol{v}_{(\infty)}$. The conjugate load power in each bus-phases can be expressed by its equivalent impedance using the solution voltage $\boldsymbol{z}_l$, i.e., $\boldsymbol{s}^* \oslash \|\boldsymbol{v}_{(\infty)}\|_{\odot}^2 = \mathbb{1} \oslash \boldsymbol{z}_l$, where $\mathbb{1}$ is a vector of ones of dimension $b\phi$. Then (6) can be reduced using Hölder's inequality as

$$\begin{aligned} \|\boldsymbol{v}_{(n+1)} - \boldsymbol{v}_{(\infty)}\|_1 &= \|\boldsymbol{B}^{-1}\,\mathbf{diag}(\alpha_p \oslash \boldsymbol{z}_l)(\boldsymbol{v}_{(n+1)} - \boldsymbol{v}_{(\infty)})\|_1 \\ &\le \|\boldsymbol{B}^{-1}\,\mathbf{diag}(\alpha_p \oslash \boldsymbol{z}_l)\|_1\|(\boldsymbol{v}_{(n+1)} - \boldsymbol{v}_{(\infty)})\|_1 \\ &\le k\|(\boldsymbol{v}_{(n+1)} - \boldsymbol{v}_{(\infty)})\|_1 \end{aligned} \tag{7}$$

From (7), for purely constant power $\alpha_p = 1$, and noticing that $\boldsymbol{B}^{-1}$ is the grid impedance matrix $\boldsymbol{Z}_{\mathrm{B}}$; then, the contraction scalar for the converged problem is

$$\begin{aligned} k &= \|\boldsymbol{Z}_{\mathrm{B}}\mathbf{diag}(\mathbb{1} \oslash \boldsymbol{z}_l)\|_1 \\ &= |\hat{z}_{jj}|/|\hat{z}_{l,j}| \end{aligned} \tag{8}$$

where the hat notation represents the solution values of the biggest ratio between impedances. The only possible solution of (8) in order to be a contraction mapping, i.e., $k < 1$, is when $|z_{jj}| < |z_{l,j}|$. The diagonal entries of $\boldsymbol{Z}_{\mathrm{B}}$, i.e., $|z_{jj}|$, are the Thevenin impedance equivalent of bus j [15], meaning that the only solution for (8) is for the operational point of the network with high impedance (high voltage, low current), which is the feasible operational state of the network. Noteworthy, $k = 0$ necessarily implies that $\alpha_p = 0$, indicating that the solution is obtained after one iteration if loads are modeled as a combination of constant current and constant impedances.

The physical interpretation of (8) can be shown by analysis of an equivalent two-bus system, discussed in detail in the next section.

3. Geometric interpretation of the existence of a power flow solution

Consider a grid formed by two nodes: node 0 as a source node, acting as the slack bus with known voltage v_0 and reference angle $\theta_0 = 0$, and node 1 with load s_l, operating as a load bus. Then, the power of the load is given by $s_l = v_l i_l^*$, defining $v_l = (z_l/(z_s + z_l))v_0$, $i_l = (v_0/(z_s + z_l))$, where the source load is $z_s = r_s + \boldsymbol{j}x_s$, and the

[1] It must be noted that voltage-controlled buses (PV-buses) are not considered in the current formulation since generators with automatic voltage regulation are not common in distribution systems [12]. The interested reader is referred to [13] if PV-buses need to be included.

[2] Public repository of the TensorPowerFlow tool is available here [14].

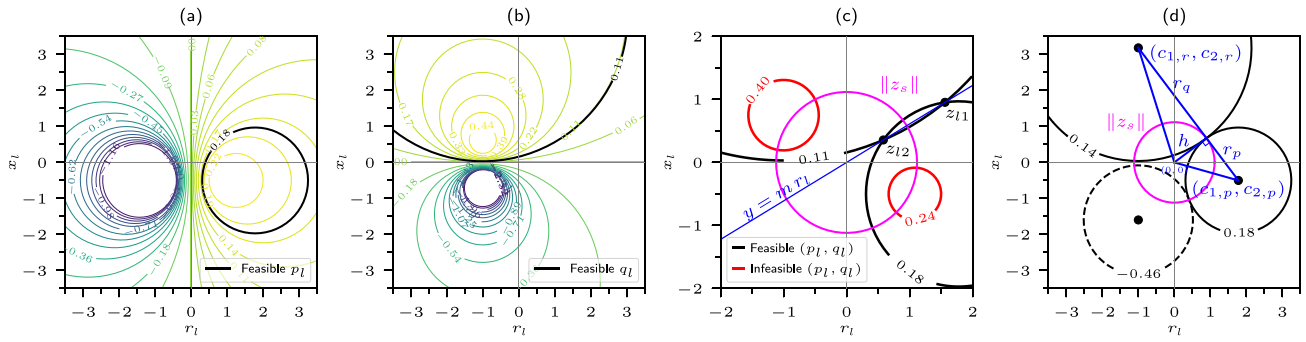


Fig. 1. Geometry of the solutions of the power flow problem for a two-bus system with parameters $z_s = 1.0 + j0.5$ and $\|v_o\| = 1.0$. (a) and (b) are the contour graphs for the circular formulation (10) and (11), for p_l and q_l, respectively; black circles highlight a feasible combination of active and reactive power. (c) The red circles highlight an infeasible value of the load power (no crossing between the circles), while the feasible values have two points of solution that form a line that passes through the origin. (d) All the solutions for the critical load power, which has only one point of contact between the circles, form a circle with radius $\|z_s\|$.

equivalent load impedance is $z_l = r_l + jx_l$. The power on the load defined in terms of the source voltage and impedances is

$$s_l = p_l + jq_l = \left(\frac{r_l}{a^2+b^2} + j\frac{x_l}{a^2+b^2}\right)\|v_0\|^2, \tag{9}$$

where $a = (r_s + r_l)$ and $b = (x_s + x_l)$. Rearranging the expressions for p_l and q_l into circle equations as (10) and (11) respectively, we have that

$$(r_l - c_{1,p})^2 + (x_l - c_{2,p})^2 = r_p^2 \tag{10}$$

$$(r_l - c_{1,q})^2 + (x_l - c_{2,q})^2 = r_q^2 \tag{11}$$

with,

$$c_{1,p} = \frac{\|v_0\|^2}{2p_l} - r_s; \quad c_{1,q} = -r_s; \quad c_{2,p} = -x_s;$$

$$c_{2,q} = \frac{\|v_0\|^2}{2q_l} - x_s; \quad r_p = \frac{\|v_0\|}{2p_l}\sqrt{\|v_o\|^2 - 4r_s p_l};$$

$$r_q = \frac{\|v_0\|}{2q_l}\sqrt{\|v_o\|^2 - 4x_s q_l}.$$

The contour plots of (10) and (11) are shown in Fig. 1(a,b), highlighting as an example the circles for the power values $p_l = 0.18$ [p.u] and $q_l = 0.11$ [p.u]. The intersections of the circles are the two possible impedance solutions for z_l, i.e., $z_{l,1}$ and $z_{l,2}$, where the values for p_l and q_l can exist simultaneously. For example, Fig. 2(c) shows two cases: the first is where the power flow has a solution and the circles intersect (black line) at two points, and the second is where the red circle does not have an intersection, meaning an infeasible combination of power values. The intersection points of the circles form a line that passes through the origin in all cases. This can be shown using the intercept in the line equation $y = mr_l + \beta$, shown as a blue line in Fig. 1(c). The intercept is defined in terms of the parameters of the two circles as

$$\beta = \frac{\overbrace{(c_{1,p}^2 + c_{2,p}^2 + r_q^2)}^{\gamma_0} - \overbrace{(c_{1,q}^2 + c_{2,q}^2 + r_p^2)}^{\gamma_1}}{2(c_{2,p} - c_{2,q})} \tag{12}$$

where, after the corresponding algebraic manipulations, it can be seen that the numerator of (12) is zero, i.e., $\gamma_0 = \gamma_1$.

The maximum loading scenario occurs when the two circles are tangent to each other (Fig. 1(d)). The tangent line of the point of contact between the circles is perpendicular to their radii and passes through their origins according to (12). Therefore, the altitude (h) of the scalene triangle formed by the centers of the circles and the origin, that is, points $\langle (c_{1,p}, c_{2,p}), (c_{1,q}, c_{2,q}), (0,0) \rangle$, can be calculated using the triangle inequalities as

$$h^2 = (c_{1,p})^2 + (c_{2,p})^2 - r_p^2 = r_s^2 + x_s^2 = \|z_s\|^2. \tag{13}$$

This means that all unique points of contact between the two circles lie in another circle defined by $\|z_s\|$, which is the magenta circle shown in Fig. 1(c,d). This also means that the critical points where the power flow is feasible and with only one impedance solution z_l, $z_{l,1} = z_{l,2}$, are when $\|z_s\| = \|z_l\|$, which is the point of maximum power transfer. From this, it is clear and *important to notice* that there are two possible solutions for the two-bus system that are not critical; one lives inside and the other outside of the circle defined by $\|z_s\|$, meaning that $\|z_{l,1}\| < \|z_s\|$ and $\|z_{l,2}\| > \|z_s\|$.

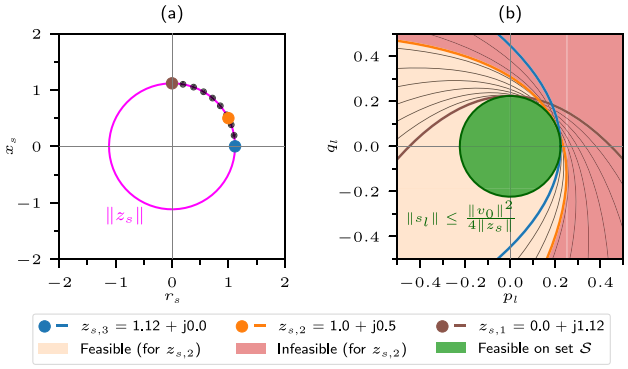


Fig. 2. Example of feasibility regions depicted by (15) and (16). (a) Each point of the maximum power transfer circle $\|z_s\|$ in the impedance plane (r_s, x_s) parameterizes the parabola in (15) that defines the feasible load power values. (b) The vertexes of the d parabolas are rotating around the circle defined by (16) in the load power plane (p_l, q_l). Only points of the first quadrant are shown, highlighting three example points $(z_{s,1}, z_{s,2}, z_{s,3})$ with their respective parabolas. Feasibility region for the case of $z_{s,2}$ is emphasized with red and yellow. The green region is the union of all possible feasible regions for all parabolas.

A region of convergence in the complex power plane (p_l, q_l) can be defined for the power in the load s_l using the critical point of contact of the two circles in the impedance plane (r_l, x_l). When the circles have one point of contact, then the distance between the centers of the circles equals the sum of its radii, that is,

$$r_p + r_q = \sqrt{(c_{1,p} - c_{1,q})^2 + (c_{2,p} - c_{2,q})^2} \tag{14}$$

Rearranging this expression, we have the following quadratic equation, named $f_\alpha(p_l, q_l)|_{(r_s, x_s)}$, for p_l and q_l

$$Gq_l^2 + Hp_l q_l + Ip_l^2 + Jq_l + Kp_l + L = 0 \tag{15}$$

where,

$$G = r_s^2 \qquad H = -2r_s x_s \qquad I = \|v_0\|^2 x_s$$
$$J = x_s^2 \qquad K = \|v_0\|^2 r_s \qquad L = -\|v_0\|^2/4$$

The conic section defined by (15) is a parabola due to its discriminant $\Delta = H^2 - 4GI = 0$. The parabola is parameterized by the source impedance z_s, i.e., r_s and x_s. An example of this parabola where z_s has

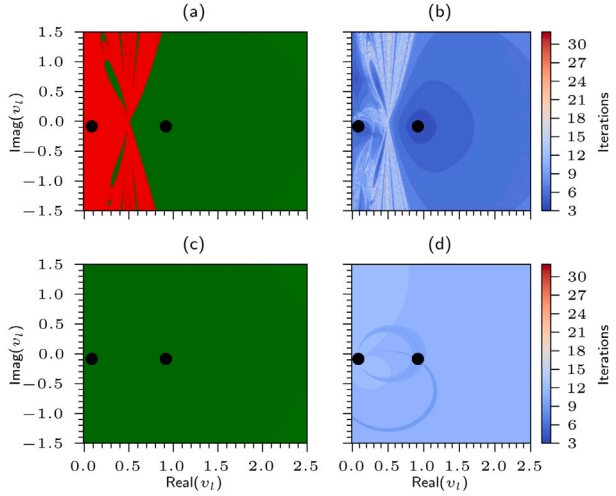


Fig. 3. Convergence analysis of NR and FPI algorithms. (a) and (c) corresponds to the regions of convergence for different initial values of voltages v_0 for NR and FPI algorithms, respectively. The green region corresponds to the high voltage (high impedance) and the red region to the low voltage (low impedance) solutions. (b) and (d) correspond to the number of iterations required for convergence for the NR and FPI algorithms.

only a resistive value is shown in blue in Fig. 2, represented by $z_{s,3}$. The parabola curve represents all (p_l, q_l) pair of values for which the two-bus system is at the maximum power transfer point, which means that it has only one equivalent load impedance solution. The region defined inside the parabola represents the power values that have a valid (feasible) solution, meaning that $z_{l,1}$ and $z_{l,2}$ exist, and the ones outside are values for which the solution of the system does not converge, i.e., it is infeasible. From Fig. 2, it can be seen that the feasible/infeasible regions change with the parameterized values of the parabola. If we compute all sets of parabolas $\mathcal{S} = \{f_\alpha(p_l, q_l)|_{(r_s,x_s)} \mid r_s^2 + x_s^2 = \|z_s\|^2\}$, and calculate the union of all feasible power flow regions defined by $\mathcal{S}$, we have the circle delimiting such a region by $\|s_l\| \leq \frac{\|v_0\|^2}{4\|z_s\|}$. This circle is the collection of all the vertices of the rotating parabolas. This means that the condition to have a feasible solution for the power flow is

$$\|v_0\|^2 \geq 4\|s_l\|\|z_s\|. \tag{16}$$

This condition that is in the norm-form, is used to prove the existence of the power flow solution in [16–18], and here it is shown as a geometrical derivation and interpretation. It should be noted that this region (16) is a *conservative estimate*, since the actual regions depend on specific parameterization of r_s and x_s [18], as shown in Fig. 2(b) for the $z_{s,2}$ example. In other words, it can be stated that a solution is feasible within the green circle; however, if it is outside the circle, it does not necessarily mean it is infeasible.

Therefore, if the power consumption satisfies at least the condition (16) [17], and the power flow converges using the iterative fixed point algorithm in (2), and following the condition in (8), then, this implies that the unique point of convergence is the high-impedance solution.

The two-bus system is solved using the NR and FPI algorithms, with different starting points $\boldsymbol{v}_0$ to numerically confirm the unique convergence point of (2). The first row of subplots in Fig. 3 corresponds to the NR and the second, to FPI solutions. The green and red colors in Fig. 3(a,c) represent the attraction regions for high-voltage (high-impedance) and low-voltage solutions, respectively. The number of iterations for each algorithm is shown as colored contour plots in Fig. 3(b,d) for the NR and FPI algorithms, respectively. The results confirm the robustness of the FPI algorithm, which converges to the operational point regardless of the initial voltage estimate, unlike NR. Notice that for a simple two-bus system, the risk of falling into a nonoperational point (but mathematically valid) still exists with the NR algorithm, and it is dependent on the $\boldsymbol{v}_0$ starting point. Additionally, the FPI algorithm has a consistent number of iterations in the complex domain for v_l, which is related to the rate of convergence determined by the contraction scalar k. The interested reader is referred to [11] for further numerical comparisons.

The NR algorithm in polar coordinates needs fewer iterations than the FPI approach [11,19]. Moreover, NR benefits from system sparsity, making it ideal for sequential computers. Still, its per-iteration computation cost is significantly higher due to the need for Jacobian inverse calculations. When dealing with hundreds to millions of power flows, the cumulative cost of NR iterations adds up, leading to longer processing times. Thanks to advancements in computer hardware, especially faster matrix multiplications, the FPI algorithm is an excellent choice for extensive power flow simulations on modern computers for distribution networks. Its reliance on successive matrix multiplications and its simple formulation makes it easy to program. Moreover, it can naturally extend to a tensor setting.

4. Multidimensional fixed point power flow

4.1. Tensor power flow - dense formulation

Consider a study that requires the analysis of an extensive number of cases of load consumption. A tensor of the power can be built as $\boldsymbol{S} \in \mathbb{C}^{\cdots \times p \times r \times t \times b\phi}$, where p, r, and t could mean a number of experiments, scenarios, and time steps, with the possibility of extending the tensor to more dimensions. The FPI algorithm in (2) in its tensor form is described as

$$\boldsymbol{V}_{(n+1)} = \boldsymbol{\mathcal{F}} \boldsymbol{V}_{(n)}^{*\,\circ(-1)} + \boldsymbol{\mathcal{W}} \tag{17}$$

where dimensions of the tensors are $\boldsymbol{\mathcal{F}} \in \mathbb{C}^{\cdots \times p \times r \times t \times b\phi \times b\phi}$, $\boldsymbol{V}_{(n+1)}$, $\boldsymbol{V}_{(n)}^{*\,\circ(-1)}, \boldsymbol{\mathcal{W}} \in \mathbb{C}^{\cdots \times p \times r \times b\phi \times t}$. An example of the structure of the tensors is shown in Fig. 4(a). Recall that the submatrices $\boldsymbol{F}$ in (3), which form the tensor $\boldsymbol{\mathcal{F}}$, are composed by a matrix multiplication that involves $\boldsymbol{Z}_B$, meaning that $\boldsymbol{\mathcal{F}}$ is dense. It should be noted that if topology does not change and the load models is constant load ($\boldsymbol{\alpha}_P = \mathbb{1}$), building a tensor $\boldsymbol{\mathcal{F}}$ only requires the repetition of the constant matrix $\boldsymbol{Z}_B$. Furthermore, with fixed voltage in the distribution transformer, the tensor $\boldsymbol{\mathcal{W}}$ is also constant and is the duplication of the vector $\boldsymbol{w}$ for all the cases under study. For the sake of compactness, the repetitions of the matrix/vector, and for generalization for any number of dimensions, we define the *dimensional tensor elements*, τ, as $\tau = \cdots \times p \times r \times t$, in order to reshape the tensors in (17) to simplify programming and enhance mathematical notation clarity.

Algorithm 1 : Tensor power flow - Dense

1: $b\phi$ = number of bus-phases, τ = dimensional tensor elements (number of power flows).
2: Input parameters: $\dot{\boldsymbol{S}} \in \mathbb{C}^{b\phi\times\tau}$, $\boldsymbol{Z}_B \in \mathbb{C}^{b\phi\times b\phi}$, $\boldsymbol{W} \in \mathbb{C}^{b\phi\times 1}$, tolerance, iterations.
3: Output parameters: $\dot{\boldsymbol{V}}_n \in \mathbb{C}^{b\phi\times\tau}$, n.
4: $\dot{\boldsymbol{V}}_0 = \mathbb{1} + j\mathbb{0}$, $\dot{\boldsymbol{V}}_0 \in \mathbb{C}^{b\phi\times\tau}$; $n = 0$; tol $= \infty$
5: **while** tol $\geq$ tolerance **and** $n <$ iterations **do**
6: **for** $i = 1$ to τ **do**
7: $\dot{\boldsymbol{V}}_{(n+1)}[i] = \boldsymbol{Z}_B(\dot{\boldsymbol{S}}[i] \odot \dot{\boldsymbol{V}}_{(n)}^{\circ(-1)}[i])^* + \boldsymbol{W}$
8: ▷ Iterate over the dimension τ of $\dot{\boldsymbol{S}}$, and $\dot{\boldsymbol{V}}_{(n)}$.
9: **end for**
10: tol $= \max(\|\boldsymbol{s}_{(n+1)} - \boldsymbol{s}_{(n)}\|^2)$; $n = n + 1$
11: **end while**
12: **return** $(\dot{\boldsymbol{V}}_{(n)}, n)$

The reshaped form of the power and voltage tensors in their two-dimensional matrix form is shown in Fig. 4(b). Their dimensions are $\dot{\boldsymbol{S}}^*, \dot{\boldsymbol{V}}_{(n+1)}, \dot{\boldsymbol{V}}^*_{(n)} \in \mathbb{C}^{b\phi\times\tau}$, where the dot notation stands for the reshaped version of the tensors. The power matrix $\dot{\boldsymbol{S}}^*$ is the concatenation of the power vectors $\boldsymbol{s}^*$ for all the cases under study along the secondary axis. It should be mentioned that reshaping the tensor does not invalidate the convergence of the FPI algorithm, as the update of the voltage values is the same as (2). The advantage of the tensor form is that matrix operations can be accelerated via parallelization on the CPU using standard low-level basic linear algebra subprograms (BLAS), e.g. OpenBLAS, LAPACK, IntelMKL, or exploiting the use of the multicore architectures from GPUs, which are specifically optimized for the parallel processing matrix multiplications. Algorithm 1 shows the implementation of the re-shaped version of (17).[3]

Although the dense formulation is simple to implement, the tensor $\boldsymbol{\mathcal{F}}$ could take a considerable amount of memory because it is dense. Therefore, for cases of networks with large $b\phi$ the sparse formulation is proposed.

4.2. Tensor power flow - Sparse formulation

The tensor algorithm in (17) can be reformulated to exploit the sparsity of $\boldsymbol{Y}_{dd}$ and $\boldsymbol{Y}_{ds}$. The sparse formulation is defined as

$$\boldsymbol{\mathcal{M}}\boldsymbol{V}_{(n+1)} = \boldsymbol{V}^{*\,\circ(-1)}_{(n)} + \boldsymbol{\mathcal{H}}. \tag{18}$$

where $\boldsymbol{\mathcal{M}} \in \mathbb{C}^{\cdots\times p\times r\times t\times b\phi\times b\phi}$ is the tensor containing the resulting tensor operation of $\boldsymbol{\mathcal{M}} = -\boldsymbol{\mathcal{A}}^{\circ(-1)}\boldsymbol{\mathcal{B}}$, where $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \in \mathbb{C}^{\cdots\times p\times r\times t\times b\phi\times b\phi}$ and $\boldsymbol{\mathcal{H}} \in \mathbb{C}^{\cdots\times p\times r\times b\phi\times t}$ a tensor containing the resulting tensors $\boldsymbol{\mathcal{H}} = \boldsymbol{\mathcal{A}}^{\circ(-1)}\boldsymbol{\mathcal{C}}$, where $\boldsymbol{\mathcal{C}} \in \mathbb{C}^{\cdots\times p\times r\times b\phi\times t}$. An example of (18) is shown in Fig. 4(c). The sparse formulation does not include the inverse of $\boldsymbol{Y}_{dd}$ in any form, and similarly to the dense formulation, the expression in (18) can be re-shaped in a two-dimensional matrix form to construct a sparse linear system of the type $Ax = b$ as

$$\underbrace{\dot{\boldsymbol{\mathcal{M}}}}_{A}\ \underbrace{\dot{\boldsymbol{V}}_{(n+1)}}_{x} = \underbrace{\dot{\boldsymbol{V}}^{*\,\circ(-1)}_{(n)} + \dot{\boldsymbol{\mathcal{H}}}}_{b}, \tag{19}$$

where the reshaped sparse matrix $\dot{\boldsymbol{\mathcal{M}}}$ is the diagonal concatenation of the submatrices of $\boldsymbol{\mathcal{M}}$. Vectors $\dot{\boldsymbol{\mathcal{V}}}_{(n+1)}$, $\dot{\boldsymbol{\mathcal{V}}}_{(n)}$, and $\dot{\boldsymbol{\mathcal{H}}}$ are vertically arranged. A visual example of the form of this reshape is shown in Fig. 4(d). The system in (19) can be solved iteratively by a sparse direct solver.

The sparse direct solvers have basically three steps: (i) analysis of the matrix $\dot{\boldsymbol{\mathcal{M}}}$ to reduce *fill-in* (via Cholesky, LU, or QR decomposition) and symbolic factorization, (ii) numerical factorization, and (iii) solving the system [20]. It is critical to note that the main advantage of (19) is that the sparse matrix $\dot{\boldsymbol{\mathcal{M}}}$ is constant for purely constant power load model and does not change during iterations. Therefore, steps (i) and (ii) are performed only once, in the first iteration, which significantly reduces the computing time for subsequent steps. The implementation of (19) is shown in Algorithm 2. It is worth recalling that in the NR algorithm, the sparse Jacobian matrix needs to be updated, requiring the first two steps in every iteration to calculate its inverse, increasing the computational time. Notice that rearranging the formulation from (17) to (18) does not invalidate the convergence of the algorithm, as the sequence of the voltage values, that is, $\{\boldsymbol{V}_{(n)}\}_{n=0}^{\infty}$, is still the same.

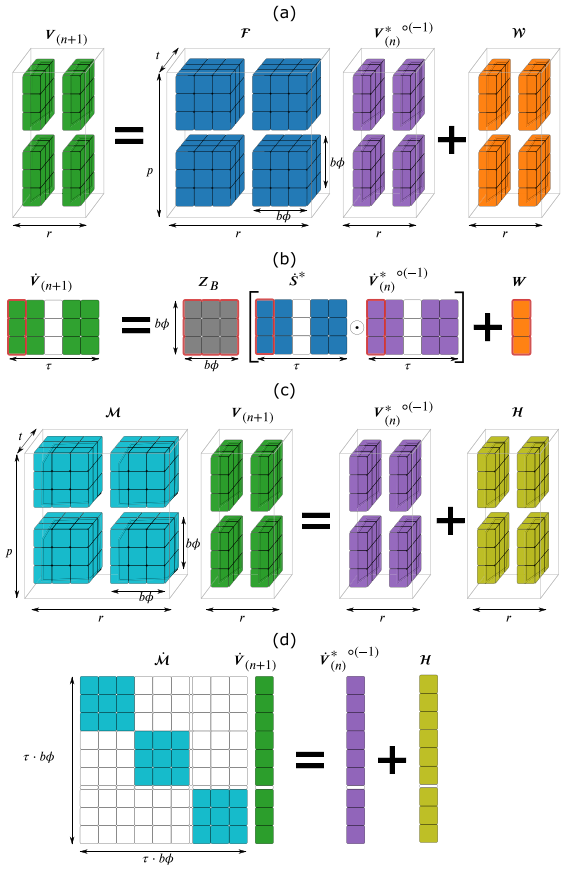


Fig. 4. Example of the tensor power flow formulations for a simulation with a four-dimensional power tensor $\boldsymbol{S} \in \mathbb{C}^{p\times r\times t\times b\phi}$, for $p = r = 2$ and $t = b\phi = 3$. (a) Visualization of the tensor dense formulation of (17). (b) In the case of constant power, the reshaped tensors use resources efficiently as the operations highlighted in red are performed concurrently. (c) Visualization of the tensor sparse formulation in (18), where tensors $\boldsymbol{\mathcal{M}}$ and $\boldsymbol{\mathcal{H}}$ are sparse. (d) Reshaped sparse formulation (19), which is solved iteratively using a direct sparse solver.

5. Simulation results

In this section, we discuss the comparison of Algorithms 1 and 2, labeled Tensor (Dense) and Tensor (Sparse), respectively, against current methods, such as SAM [11], NR with its sparse formulation in polar coordinates (NR (Sparse)) [21] and as implemented in the PandaPower package [22]; and the backward-forward sweep method (BFS) [23]. A more comprehensive comparison using different PF algorithms can be found in [11]. Additionally, the tensor-dense formulation is programmed to use a GPU to quantify computational speed improvements; this implementation is named Tensor (GPU). The implementation of Algorithms 1 and 2 is publicly available to the community as a Python

[3] Note that variable `tol` is evaluated after the first iteration to guarantee the convergence of the algorithm even if another mathematically feasible solution is chosen as starting point.

Algorithm 2 : Tensor power flow - Sparse

Input parameters:
$\dot{\boldsymbol{S}} \in \mathbb{C}^{b\phi\times\tau}$, $\boldsymbol{Y}_{dd} \in \mathbb{C}^{b\phi\times b\phi}$, $\boldsymbol{Y}_{ds} \in \mathbb{C}^{b\phi\times 1}$, $\boldsymbol{v}_s \in \mathbb{C}$, tolerance, iterations. ▷ $\boldsymbol{Y}_{dd}$ and $\boldsymbol{Y}_{ds}$ are sparse.
Output parameters: $\dot{\boldsymbol{V}}_{(n)} \in \mathbb{C}^{b\phi\times\tau}$, n.
$\dot{\mathcal{M}} = -\text{diag}(\dot{\boldsymbol{S}}^{*(-1)}_{[0]})\boldsymbol{Y}_{dd}$ ▷ [0] first entry over dimension τ.
$\dot{\mathcal{H}} = -\text{diag}(\dot{\boldsymbol{S}}^{*(-1)}_{[0]})\boldsymbol{Y}_{ds}\boldsymbol{v}_s$
for $i = 1$ to τ **do**
  $\boldsymbol{m} = -\text{diag}(\dot{\boldsymbol{S}}^{*(-1)}_{[i]})\boldsymbol{Y}_{dd}$
  $\boldsymbol{h} = -\text{diag}(\dot{\boldsymbol{S}}^{*(-1)}_{[i]})\boldsymbol{Y}_{ds}$
  $\dot{\mathcal{M}} = \left(\begin{array}{c|c} \dot{\mathcal{M}} & \mathbb{0} \\ \hline \mathbb{0} & \boldsymbol{m} \end{array}\right)$ ▷ Concat. sparse matrix diagonally
  $\dot{\mathcal{H}} = \left(\begin{array}{c} \dot{\mathcal{H}} \\ \hline \boldsymbol{h} \end{array}\right)$ ▷ Concatenate sparse vectors
end for
$\dot{\boldsymbol{V}}_0 = \mathbb{1} + \boldsymbol{j}\mathbb{0}$, $\dot{\boldsymbol{V}}_0 \in \mathbb{C}^{(b\phi\cdot\tau)\times 1}$; $n = 0$; tol $= \infty$
while tol $\geq$ tolerance **and** $n <$ iterations **do**
  $\dot{\boldsymbol{V}}_{(n+1)} = \text{solve}(\dot{\mathcal{M}}, \dot{\boldsymbol{V}}^{*o(-1)}_{(n)} + \dot{\mathcal{N}})$ ▷ Sparse system (19)
  tol $= \max(\|\boldsymbol{s}_{(n+1)} - \boldsymbol{s}_{(n)}\|^2)$; $n = n + 1$
end while
$\boldsymbol{V}_n = \text{reshape}(\dot{\boldsymbol{V}}_n, (b\phi \times \tau))$
return $(\boldsymbol{V}_{(n)}, n)$

package named TensorPowerFlow.[4] The experiments were conducted on a conventional laptop with an Intel(R) Core(TM) i7-7700HQ CPU @ 2.80 GHz, with 8 Logical Processor(s), and a NVIDIA Quadro M1200 with 4 GB (GDDR5) of memory.

5.1. Computational performance

The goal of the experiments is to test the computation performance of the different methods for two aspects: (i) grid size and (ii) dimensionality. For the first aspect, different networks with bus-phases $b\phi$ ranging from 9 to 5k are generated using a k-ary tree generative model, with random k-child between 1 and 5, in order to simulate the radial structure of the distribution system [24]. For the second aspect, the dimensional tensor elements τ range from 10 to 525k, which is equivalent to the number of PFs to be computed, and the power values are generated using a multivariate elliptical copula [25] to simulate realistic scenarios of consumption profiles. For consistency, the same power scenarios and networks are used for all methods. Additionally, for methods that use sparse solvers, the Intel oneMKL - PARDISO is used in all of them for a fair comparison.

The performance of the methods to the increase of $b\phi$ is shown for one single PF in Fig. 5(a), and for 500 PFs Fig. 5(b). The fastest methods for one single PF are SAM and Tensor (Dense), with an execution time of ≈0.1 [ms] for the smallest network. However, both methods reduce their performance as the grid size increases. Notice that around 900 bus-phases the Tensor (Sparse) formulation outperforms the SAM; around 2k bus-phases it is outperformed by Tensor (Sparse), and continues with the trend as the size continues to increase. In the case of 500 PF (Fig. 5(b)), the Tensor (Dense) continues as the top performer for smaller grids (<500 bus-phases), seconded by the Tensor (Sparse) where after a grid size of 2k is better than the Tensor (Dense). Noteworthy, the SAM lags behind from the start because it requires more matrix multiplications in its original formulation. An interesting behavior is seen for the Tensor (GPU) implementation, which is significantly slower for a small number of power flows (Fig. 5(a)); however, when the number of matrix operations is large, either due to an increase of τ or $b\phi$, the Tensor (GPU) outperforms the others, as seen in Fig. 5(b)(c). This behavior can be explained due to the overhead time required to transfer the data from the host computer to the GPU.

[4] Public repository of the TensorPowerFlow tool is available here [14].

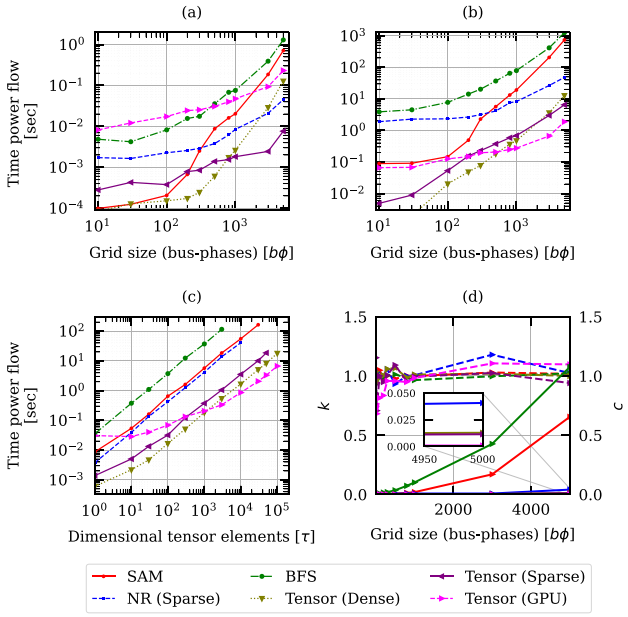


Fig. 5. Comparison of the performance of the algorithms. Computational wall-times (t_c) increase the bus-phases size, $b\phi$, for (a) 1 power flow, and (b) 500 power flows. (c) Test for increased dimensional tensor elements τ, for a grid size of 500 bus-phases ($b\phi$). (d) Computational complexity fit with asymptotic complexity model $t_c = c \cdot n^k$ (solid lines are c, dotted k). Tensor *dense*, *sparse*, and *GPU* are the proposed algorithms in this paper.

Finally, the empirical computational complexity of the algorithms has been estimated for a set of $b\phi$ values by running 17k PFs each time, using a power-law-fitting model [26]. The power-law-fitting gives a concise, quantitative way to compare the computational cost of each algorithm as the number of PF increases (τ) for each $b\phi$. That is, finding the value for k and c for $t_c = c\,n^k$ as n increases. The results are shown in Fig. 5(d). For example, for $b\phi = 3000$ the pairs $\{k, c\}$ are: SAM $\{1.03, 0.17\}$; BFS $\{0.99, 0.43\}$; NR (Sparse) $\{1.18,\ 7.79 \times 10^{-3}\}$; Tensor (Dense) $\{1.02,\ 3.21 \times 10^{-3}\}$; Tensor (Sparse) $\{1.02, 2.53 \times 10^{-3}\}$; and Tensor (GPU) $\{1.10, 0.25 \times 10^{-3}\}$. Notice that all tested methods show a close to linear complexity ($k \approx 1$) for each $b\phi$, and the value of c establishes the difference between them. This close to linear complexity means that for a given network (fixed topology), the computational time increases approximately linearly with τ, which can be graphically inferred from Fig. 4(b) for the proposed algorithms and can be extended analogously to the SAM, BFS, and NR (Sparse). On the other hand, the value of c is proportional to the complexity of a single PF for a given grid, e.g., the number of operations required. Results confirm that the SAM and BFS scale poorly as the grid size increases, as already seen in Fig. 5(b). A better performance can be seen for NR (Sparse), which is close to the three proposed tensor formulations. However, notice that c in the NR (Sparse) is at least twice as high as in the Tensor (Dense), three times higher than in the Tensor (Sparse) and thirty times higher than in Tensor (GPU). These results indicate that as the grid size increases, the preferred formulation is Tensor (GPU), followed by Tensor (Sparse) and Tensor (Dense).

5.2. Application in yearly time series simulation

A comparative test is performed for a yearly TSS study for grid sizes ranging from 100 to 5000 bus-phases ($b\phi$), for time resolution of 1, 15, 30, and 60 min. The results are shown in Table 1, where the maximum waiting time for the results is set to 9 h. In general, all the FPI methods are preferred and best suited for large amounts of PFs instead of conventional NR and BFS techniques. Among the iterative methods, the proposed Tensor (Dense) algorithm stood out as the fastest

Table 1
Computing wall-times for solving one year of power flows at different time resolutions (table values in minutes).

Algorithm	Grid size ($b\phi$)	@1 h (8760 pf)	@30 min (17 520 pf)	@15 min (35 040 pf)	@1 min (525 600 pf)
BFS	100	1.98	3.96	7.91	118.71
NR (Sparse)	100	1.03	2.06	4.12	61.85
Tensor (Sparse)	100	0.17	0.34	0.68	10.14
SAM	100	0.02	0.04	0.09	1.29
Tensor (GPU)	100	0.02	0.04	0.09	1.30
Tensor (Dense)	100	**0.01**	**0.01**	**0.02**	**0.37**
BFS	5000	186.49	372.99	745.97	–
NR (Sparse)	5000	8.77	17.54	35.08	526.23
Tensor (Sparse)	5000	2.43	4.86	9.71	145.71
SAM	5000	108.09	216.18	432.36	–
Tensor (GPU)	5000	**0.92**	**1.84**	**3.69**	**55.34**
Tensor (Dense)	5000	2.60	5.19	10.38	155.71

for the smaller grid size case. It completed the 1-minute resolution case (approximately 525k PFs calculations) in only 22 s. To put this in perspective, compared to the NR (Sparse) method – which took more than one hour to accomplish the same task – the Tensor (Dense) displayed a remarkable speedup factor of 164.

In the case of larger grids (5k bus-phases), the proposed methods are still more efficient than NR (Sparse), and the Tensor (Sparse) is the second best for all time resolutions. However, the Tensor (GPU) is the fastest, showing the speedup of using GPU, which is capable of massively parallelizing the matrix multiplications in a multi-thread setting. It is worth mentioning that a GPU-based sparse solver could also be utilized to enhance the Tensor (Sparse) formulation, e.g., using STRUMPACK or MAGMA [27], though this area requires further exploration.

6. Remarks and potential applications

The obtained results indicate that as the grid size increases, the preferred formulation is Tensor (GPU), followed by Tensor (Sparse) and Tensor (Dense). It is important to note that performance-tuning strategies, such as reducing data transmission by data reuse, optimizing task allocation, and optimizing memory access, could improve the performance of multithreading tasks, which might further improve the use of GPUs in the proposed algorithms. Also, since the Tensor (Dense) relies on calculating the impedance matrix, the formulation loses performance if the system's topology changes over the dimensional tensor elements, e.g., reconfiguration and tap-changes. Despite this, speedup factors up to 164 times compared to the NR (Sparse) formulation were evidenced for grids of 5k bus-phases and more than 500k PFs with the proposed algorithms, highlighting its potential application on a wide range of analyses in distribution systems. Here we highlight some of them:

Probabilistic analyses

Exogenous uncertainties can be discretized and mapped using scenarios to be later evaluated using a power flow formulation. This is already a computationally demanding task for a single period, as the number of scenarios is typically on the order of tens of thousands. Complexity increases when considering more than one period, such as to account for temporal correlations over a day or a whole year. An example of these problems is the *multi-period probabilistic power flow*. The SAM algorithm was used in [28] to run more than 7 million PFs (dimensional tensor elements τ) in a network of $b\phi = 34$ to evaluate the risk of technical violations resulting from high photovoltaic penetration in distribution networks. The assessment required the use of a high-performance computing unit for over a week. Given the grid size and the number of dimensional tensor elements involved, this scenario would be a perfect application for the Tensor (Dense) or the Tensor (GPU) formulations. Other examples may include assessing the impacts of electric vehicles and electrification of heating systems, demand response and congestion management, and resilience assessment, among others.

Machine learning

Machine learning (ML) algorithms rely on a large number of data samples during training, e.g., supervised, unsupervised learning and reinforcement learning (RL). The proposed PF algorithms could be integrated into RL environments that require solving a large number of PF formulations to train RL agents aiming to learn optimal control policies. An example of such an application can be found in [3], where an RL was used for community battery operations, requiring thousands of PFs to evaluate the reward of each action. A similar work is done in [29] to enforce voltage magnitude limits due to high PV penetration using decentralized RL agents. In these cases, the dimensional tensor elements would be the number of RL agents, the number of time periods, and the number of actions. The proposed PF formulations can also be used to train surrogate PF models, for instance, those based on deep learning-based models. This includes models such as graph neural networks (GNNs) that exploit the natural graph structure of the power system to accelerate PF [30] and state estimation calculations [31]. Other applications might include network resiliency and assessment by integrating probabilistic risk models with ML-based decision systems [32].

Optimization

Optimization problems considering the network rely on efficient power flow formulations. Specifically, when using metaheuristic algorithms, hundreds of thousands of PFs are needed, whether they are linked to particles, agents, offspring, swarms, leaders, states, or any other description of solution candidates that requires evaluating the objective function iteratively [33]. For example, the SAM algorithm was used in [34] to evaluate the objective function based on candidate solutions using the advanced arithmetic optimizer algorithm to obtain optimal tap positions of voltage regulators and charging patterns for energy storage devices. Using one of the proposed tensor formulations by reshaping each agent, iteration, and candidate as dimensional tensor elements would decrease the computational time compared to the used PF algorithm, allowing for a more exhaustive exploration phase and potentially improving the quality of the solution within the same execution time. Other potential metaheuristics applications include planning under uncertainty (e.g., stochastic/chance-constrained optimal power flow) and bi-level problems (e.g., market clearing, min–max problems).

7. Conclusions

This paper presented practical PF formulations for analyses of distribution systems in a multidimensional scope. The formulations are based on a FPI algorithm, and two algorithms are presented in their tensor forms for dense and sparse versions. The convergence proof of the algorithms and the existence of a solution are presented with its geometrical and physical interpretation. The mathematical analysis shows that the FPI algorithm converges to the high impedance value in the case of the existence of a solution, and it can be extended to the multidimensional formulation. The performance of the proposed algorithms compared to conventional methods such as NR and BFS was

evaluated in two aspects: the size of the grid (number of bus-phases, $b\phi$) and dimensionality (number of PFs, τ).

For smaller grids, Tensor (Dense) outperforms the other methods (<1k $b\phi$), while Tensor (Sparse) became the most efficient as the grid size increased. The algorithms were tested in a practical TSS application for different grid sizes and time resolutions, and some extra potential applications were described. Tensor (Dense) emerged as the fastest algorithm, providing a significant speedup over traditional methods like NR (Sparse), especially for smaller grid sizes. The study also introduced a GPU implementation, Tensor (GPU), which suffers from data transfer overhead for smaller grids and low τ but excelled when the number of matrix operations was substantial, as seen when the grid size and dimensionality increased. The results indicate that as the grid size increases and a large number of PFs are required, the preferred formulation is Tensor (GPU), followed by Tensor (Sparse) and Tensor (Dense).

CRediT authorship contribution statement

Edgar Mauricio Salazar Duque: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Juan S. Giraldo:** Writing – review & editing. **Pedro P. Vergara:** Writing – review & editing. **Phuong H. Nguyen:** Writing – review & editing, Supervision. **Han (J.G.) Slootweg:** Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data availability details are provided in Ref. [14].

Appendix

In the case of a non-constant power load model, where $\boldsymbol{\alpha}_{\mathrm{P}} \neq \mathbb{1}$ and $\boldsymbol{\alpha}_{\mathrm{Z}}+\boldsymbol{\alpha}_{\mathrm{I}}+\boldsymbol{\alpha}_{\mathrm{P}} = \mathbb{1}$, Algorithm 2 would require modifications, which come from accounting the full ZIP load model from (3).

The necessary changes are for lines 4–5 and 7–8:

4: $\dot{\boldsymbol{\mathcal{M}}} = -\mathrm{diag}(\boldsymbol{\alpha}_{\mathrm{P}_{[0]}} \odot \dot{\boldsymbol{S}}_{[0]}^{*(-1)})\boldsymbol{Y}_{\boldsymbol{dd}} - \mathrm{diag}((\boldsymbol{\alpha}_{\mathrm{P}_{[0]}} \odot \boldsymbol{\alpha}_{\mathrm{Z}_{[0]}}) \oslash \dot{\boldsymbol{S}}_{[0]}^{\circ(2)})$

5: $\dot{\boldsymbol{\mathcal{H}}} = -\mathrm{diag}(\boldsymbol{\alpha}_{\mathrm{P}_{[0]}} \odot \dot{\boldsymbol{S}}_{[0]}^{*(-1)})\boldsymbol{Y}_{\boldsymbol{ds}}\boldsymbol{v}_s + \mathrm{diag}(\boldsymbol{\alpha}_{\mathrm{P}_{[0]}} \odot \boldsymbol{\alpha}_{\mathrm{I}_{[0]}})$

(...)

7: $\boldsymbol{m} = -\mathrm{diag}(\boldsymbol{\alpha}_{\mathrm{P}_{[i]}} \odot \dot{\boldsymbol{S}}_{[i]}^{*(-1)})\boldsymbol{Y}_{\boldsymbol{dd}} - \mathrm{diag}((\boldsymbol{\alpha}_{\mathrm{P}_{[i]}} \odot \boldsymbol{\alpha}_{\mathrm{Z}_{[i]}}) \oslash \dot{\boldsymbol{S}}_{[i]}^{\circ(2)})$

8: $\boldsymbol{h} = -\mathrm{diag}(\boldsymbol{\alpha}_{\mathrm{P}_{[i]}} \odot \dot{\boldsymbol{S}}_{[i]}^{*(-1)})\boldsymbol{Y}_{\boldsymbol{ds}}\boldsymbol{v}_s + \mathrm{diag}(\boldsymbol{\alpha}_{\mathrm{P}_{[i]}} \odot \boldsymbol{\alpha}_{\mathrm{I}_{[i]}})$

The sparse algorithm requires additional operations to prepare the matrix $\dot{\boldsymbol{\mathcal{M}}}$ and vector $\dot{\boldsymbol{\mathcal{H}}}$ as per formulation (19). However, this preparation is performed only once (lines 1–11). Consequently, there is no significant increase in the computational time, as the power flow convergence process (lines 13–16) is expected to dominate the computational time. It should also be noted that the sizes of $\dot{\boldsymbol{\mathcal{M}}}$ and $\dot{\boldsymbol{\mathcal{H}}}$ do not increase in size when the complete ZIP load model is incorporated. Therefore, the computational complexity for the Tensor (Sparse) in Fig. 5 remains unchanged.

References

[1] Qureshi Muhammad Umer, Grijalva Santiago, Reno Matthew J, Deboever Jeremiah, Zhang Xiaochen, Broderick Robert J. A fast scalable quasi-static time series analysis method for PV impact studies using linear sensitivity model. IEEE Trans Sustain Energy 2019;10(1):301–10.

[2] Liu Chengxi, Sun Kai, Wang Bin, Ju Wenyun. Probabilistic power flow analysis using multidimensional holomorphic embedding and generalized cumulants. IEEE Trans Power Syst 2018;33(6):7132–42.

[3] Duque Edgar Mauricio Salazar, Giraldo Juan S, Vergara Pedro P, Nguyen Phuong, van der Molen Anne, Slootweg Han. Community energy storage operation via reinforcement learning with eligibility traces. Electr Power Syst Res 2022;212:108515.

[4] Garcés Alejandro, Mora Juan José, Useche Mario-Alejandro. Putting tensors back in power systems analysis. In: 2019 int. conf. on smart energy systems and technologies. SEST, 2019, p. 1–5.

[5] Nickolls John, Dally William J. The GPU computing era. IEEE Micro 2010;30(2):56–69.

[6] Zhou Gan, Bo Rui, Chien Lungsheng, Zhang Xu, Shi Fei, Xu Chunlei, Feng Yanjun. GPU-based batch LU-factorization solver for concurrent analysis of massive power flows. IEEE Trans Power Syst 2017;32(6):4975–7.

[7] Li Xue, Li Fangxing, Yuan Haoyu, Cui Hantao, Hu Qinran. GPU-based fast decoupled power flow with preconditioned iterative solver and inexact Newton method. IEEE Trans Power Syst 2017;32(4):2695–703.

[8] Zhou Gan, Bo Rui, Chien Lungsheng, Zhang Xu, Yang Shengchun, Su Dawei. GPU-accelerated algorithm for online probabilistic power flow. IEEE Trans Power Syst 2018;33(1):1132–5.

[9] Zhou Gan, Feng Yanjun, Bo Rui, Chien Lungsheng, Zhang Xu, Lang Yansheng, Jia Yupei, Chen Zhengping. GPU-accelerated batch-ACPF solution for N-1 static security analysis. IEEE Trans Smart Grid 2017;8(3):1406–16.

[10] Montoya Oscar Danilo, Gil-González Walter. On the numerical analysis based on successive approximations for power flow problems in AC distribution systems. Electr Power Syst Res 2020;187:106454.

[11] Giraldo Juan S, Montoya Oscar Danilo, Vergara Pedro P, Milano Federico. A fixed-point current injection power flow for electric distribution systems using Laurent series. Electr Power Syst Res 2022;211:108326.

[12] Petinrin JO, Shaabanb Mohamed. Impact of renewable generation on voltage control in distribution systems. Renew Sustain Energy Rev 2016;65:770–83.

[13] Garcia Paulo AN, Pereira Jose Luiz R, Carneiro Sandoval, Da Costa Vander M, Martins Nelson. Three-phase power flow calculations using the current injection method. IEEE Trans Power Syst 2000;15(2):508–14.

[14] Duque EM Salazar. Tensorpowerflow. 2024, https://github.com/MauricioSalazare/tensorpowerflow.

[15] Grainger John J, Stevenson William D. Power system analysis. McGraw-Hill series in electrical and computer engineering power and energy, New York, NY St. Louis San Francisco: McGraw-Hill, Inc; 1994, p. 289.

[16] Sur Ujjal, Sarkar Gautam. Existence of explicit and unique necessary conditions for power flow insolvability in power distribution systems. IEEE Syst J 2019;13(1):702–9.

[17] Bolognani Saverio, Zampieri Sandro. On the existence and linear approximation of the power flow solution in power distribution networks. IEEE Trans Power Syst 2016;31(1):163–72.

[18] Yu Suhyoun, Nguyen Hung D, Turitsyn Konstantin S. Simple certificate of solvability of power flow equations for distribution systems. In: 2015 IEEE power & energy society general meeting. Denver, CO, USA: IEEE; 2015, p. 1–5.

[19] Ahmadi Afshin, Smith Melissa C, Collins Edward R, Dargahi Vahid, Jin Shuangshuang. Fast Newton-Raphson power flow analysis based on sparse techniques and parallel processing. IEEE Trans Power Syst 2022;37(3):1695–705.

[20] Davis Timothy A. Direct methods for sparse linear systems. Fundamentals of algorithms, Philadelphia: SIAM; 2006.

[21] Sereeter Baljinnyam, Vuik Cornelis, Witteveen Cees. On a comparison of Newton–Raphson solvers for power flow problems. J Comput Appl Math 2019;360:157–69.

[22] Thurner Leon, Scheidler Alexander, Schafer Florian, Menke Jan-Hendrik, Dollichon Julian, Meier Friederike, Meinecke Steffen, Braun Martin. Pandapower—An open-source python tool for convenient modeling, analysis, and optimization of electric power systems. IEEE Trans Power Syst 2018;33(6):6510–21.

[23] Bompard E, Carpaneto E, Chicco G, Napoli R. Convergence of the backward/forward sweep method for the load-flow analysis of radial distribution systems. Int J Electr Power Energy Syst 2000;22(7):521–30.

[24] Storer James A. An introduction to data structures and algorithms. Boston, MA: Birkhäuser Boston; 2002, p. 225.

[25] Duque Edgar Mauricio Salazar, Vergara Pedro P, Nguyen Phuong H, Van Der Molen Anne, Slootweg JG. Conditional multivariate elliptical copulas to model residential load profiles from smart meter data. IEEE Trans Smart Grid 2021;12(5):4280–94.

[26] Goldsmith Simon F, Aiken Alex S, Wilkerson Daniel S. Measuring empirical computational complexity. In: Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering. 2007, p. 395–404.

[27] Ghysels Pieter, Synk Ryan. High performance sparse multifrontal solvers on modern GPUs. Parallel Comput 2022;110:102897.
[28] Duque Edgar Mauricio Salazar, Giraldo Juan S, Vergara Pedro P, Nguyen Phuong H, Molen Anne van der, Slootweg JG. Risk-Aware Operating Regions for PV-rich distribution networks considering irradiance variability. IEEE Trans Sustain Energy 2023;14(4):2092–108.
[29] Vergara Pedro P, Salazar Mauricio, Giraldo Juan S, Palensky Peter. Optimal dispatch of PV inverters in unbalanced distribution systems using Reinforcement Learning. Int J Electr Power Energy Syst 2022;136:107628.
[30] Lin Nan, Orfanoudakis Stavros, Cardenas Nathan Ordonez, Giraldo Juan S, Vergara Pedro P. PowerFlowNet: Power flow approximation using message passing Graph Neural Networks. Int J Electr Power Energy Syst 2024;160:110112.
[31] Habib Benjamin, Isufi Elvin, Breda Ward van, Jongepier Arjen, Cremer Jochen L. Deep statistical solver for distribution system state estimation. IEEE Trans Power Syst 2024;39(2):4039–50.
[32] Konstantelos Ioannis, Sun Mingyang, Tindemans Simon H, Issad Samir, Panciatici Patrick, Strbac Goran. Using vine copulas to generate representative system states for machine learning. IEEE Trans Power Syst 2019;34(1):225–35.
[33] Ahmadi Bahman, Giraldo Juan S, Hoogsteen Gerwin. Dynamic hunting leadership optimization: algorithm and applications. J Comput Sci 2023;69:102010.
[34] Ahmadi Bahman, Giraldo Juan S, Hoogsteen Gerwin, Gerards Marco ET, Hurink Johann L. A multi-objective decentralized optimization for voltage regulators and energy storage devices in active distribution systems. Int J Electr Power Energy Syst 2023;153:109330.