Chapter

Solution Methods of Large Complex-Valued Nonlinear System of Equations

Robson Pires

Abstract

Nonlinear systems of equations in complex plane are frequently encountered in applied mathematics, e.g., power systems, signal processing, control theory, neural networks, and biomedicine, to name a few. The solution of these problems often requires a first- or second-order approximation of nonlinear functions to generate a new step or descent direction to meet the solution iteratively. However, such methods cannot be applied to functions of complex and complex conjugate variables because they are necessarily nonanalytic. To overcome this problem, the Wirtinger calculus allows an expansion of nonlinear functions in its original complex and complex conjugate variables once they are analytic in their argument as a whole. Thus, the goal is to apply this methodology for solving nonlinear systems of equations emerged from applications in the industry. For instances, the complex-valued Jacobian matrix emerged from the power flow analysis model which is solved by Newton-Raphson method can be exactly determined. Similarly, overdetermined Jacobian matrices can be dealt, e.g., through the Gauss-Newton method in complex plane aimed to solve power system state estimation problems. Finally, the factorization method of the aforementioned Jacobian matrices is addressed through the *fast Givens transformation* algorithm which means the square root-free Givens rotations method in complex plane.

Keywords: large nonlinear system of equation solution in complex plane, complex-valued Newton-Raphson and gauss-Newton iterative algorithms, Cartesian coordinates

1. Introduction

This work is a tribute to Steinmetz's contribution [1]. The reasons and motivations are stated throughout the whole document once the numerical solutions for solving power system applications are typically carried out in the real domain. For instance, the power flow analysis and power system state estimation are well-known tools, among others. It turns out that these solutions are not well suited for modeling voltage and current phasor. To overcome this difficulty, the proposal described in this chapter aims to model the aforementioned applications in a unified system of coordinates, e.g., complex domain. Nonetheless, the solution methods of these problems often require a first- or second-order approximation of the set of power flow equations; such methods cannot be applied to nonlinear functions of

complex variables because they are nonanalytic in their arguments. Consequently, for these functions Taylor series expansions do not exist. Hence, for many decades this problem has been solved redefining the nonlinear functions as separate functions of the real and imaginary parts of their complex arguments so that standard methods can be applied. Although not widely known, it is also possible to construct an extended nonlinear function that includes not only the original complex state variables but also their complex conjugates, and then the Wirtinger calculus can be applied [2]. This property lies on the fact that if a function is analytic in the space spanned by $\Re\{x\}$ and $\{x\}$ in $\mathbb{R}$, it is also analytic in the space spanned by x and x^* in $\mathbb{C}$. In complex analysis of one and several complex variables, *Wirtinger operators* are partial differential operators of the first order which behave in a very similar manner to the ordinary derivatives with respect to one real variable, when applied to *holomorphic* functions, *non-holomorphic* functions, or simply differentiable functions on complex domain. These operators allow the construction of a differential calculus for such functions that is entirely analogous to the ordinary differential calculus for functions of real variables [2, 3]. Then, taken into account the Wirtinger calculus, this chapter shows how the Jacobian matrix patterns emerge in complex plane corresponding to the steady-state models of power flow analysis and power system state estimation, respectively.

In this chapter the classical Newton-Raphson and Gauss-Newton methods in complex plane aiming the numerical solution of the power flow analysis and power system state estimation are derived, respectively. Moreover, the factorization methods addressed to deal with the Jacobian matrices emerged from these approaches are included [4].

This chapter is organized as follows. The theoretical foundation which is based on *Wirtinger calculus* is summed up in Section 2. Section 3 describes two algorithms suggested to factorize Jacobian matrix in complex plane regardless if it is exactly determined or overdetermined. In Section 4, the complex-valued static model solution by using Newton-Raphson method is derived, whereas in Section 5, the Gauss-Newton method developed in complex plane is equally presented. Finally, in Section 6 some conclusions are gathered and stated the next issues to be investigated in the near future.

2. Theoretical foundation

2.1 Complex differentiability

A complex function is defined as

$$f(x) = u(a,b) + j\,v(a,b), \tag{1}$$

where $x = a + j\,b$ and $u(a,b)$, $v(a,b)$ are real functions, u, v: $\mathbb{R}^2 \to \mathbb{R}$. Functions like Eq. (1) are in general complex but may be real-valued in special cases, e.g., squared error cost function $\mathcal{J}\left(\left|\mathbf{e}^2\right|\right)$. The definition of complex differentiability requires that the derivatives defined as the limit be independent of the direction in which Δx approaches 0 in complex plane:

$$f'(x_0) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}. \tag{2}$$

This requires that the Cauchy-Riemann equations be satisfied, i.e.,

$$\frac{\partial u}{\partial a} = \frac{\partial v}{\partial b}, \qquad \frac{\partial v}{\partial a} = -\frac{\partial u}{\partial b}. \tag{3}$$

These conditions are necessary for $f(x)$ to be complex differentiable. If the partial derivatives of $u(a,b)$ and $v(a,b)$ are continuous on their entire domain, then they are sufficient as well. Therefore, the complex function $f(x)$ is called an analytic or holomorphic function [2]. As an example, let $f(x) = x^2$ be a complex function with $x = a + j\,b$. Then,

$$f(x) = x^2 = \underbrace{a^2 - b^2}_{=u} + j\,\underbrace{2ab}_{=v} = y, \tag{4}$$

which under differentiation rule leads to

$$\frac{\partial u}{\partial a} = 2a = \frac{\partial v}{\partial b} = 2a; \qquad \frac{\partial u}{\partial b} = -2b = -\left(\frac{\partial v}{\partial a} = 2b\right). \tag{5}$$

These results show that the Cauchy-Riemann equations hold, and hence $f(x) = y = x^2$ is a holomorphic function.

2.2 *CR-Calculus* or Wirtinger calculus

Introduced by Wilhelm Wirtinger in 1927 [2], the *CR-Calculus*, also known as the Wirtinger calculus, provides a way to differentiate nonanalytic functions of complex variables. Specifically, this calculus is applicable to a function $f(x)$ given by Eq. (1) if $u(a,b)$ and $v(a,b)$ have continuous partial derivatives with respect to a and b, yielding

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial a}\frac{\partial a}{\partial x} + \frac{\partial f}{\partial b}\frac{\partial b}{\partial x}. \tag{6}$$

Since we have

$$a = \frac{(x + x^*)}{2}, \qquad \partial a = \frac{(\partial x + \partial x^*)}{2}, \tag{7}$$

$$b = j\,\frac{(x^* - x)}{2}, \qquad \partial b = j\,\frac{(\partial x^* - \partial x)}{2}, \tag{8}$$

and by setting $\frac{\partial x^*}{\partial x}$ to zero, it follows that

$$\frac{\partial f}{\partial x} = \frac{1}{2}\left(\frac{\partial f}{\partial a} - j\,\frac{\partial f}{\partial b}\right). \tag{9}$$

Note that the Cauchy-Riemann conditions for $f(\cdot)$ to be analytic in x can be expressed compactly using the gradient as $\frac{\partial f}{\partial x^*} = 0$, i.e., $f(\cdot)$ is a function of only x.

Similarly, if we take the derivative of $f(\cdot)$ with respect to x^*, that is,

$$\frac{\partial f}{\partial x^*} = \frac{\partial f}{\partial a}\frac{\partial a}{\partial x^*} + \frac{\partial f}{\partial b}\frac{\partial b}{\partial x^*}. \tag{10}$$

By setting $\frac{\partial x}{\partial x^*}$ to zero, we get

$$\frac{\partial f}{\partial x^*} = \frac{1}{2}\left(\frac{\partial f}{\partial a} + j\,\frac{\partial f}{\partial b}\right). \tag{11}$$

Again, the Cauchy-Riemann conditions for $f(\cdot)$ to be analytic in x^* can be expressed compactly using the gradient as $\frac{\partial f}{\partial x} = 0$, i.e., $f(\cdot)$ is a function only of x^*.

In other words, the gradient (respectively conjugate gradient) operator acts as a partial derivative with respect to x (respectively to x^*), treating x^* (respectively x) as a constant. Formally, we have

$$\frac{\partial f(x_c)}{\partial x} = \left.\frac{\partial f(x,x^*)}{\partial x}\right|_{x^*=Const} = \frac{1}{2}\left(\frac{\partial f}{\partial a} - j\,\frac{\partial f}{\partial b}\right), \tag{12}$$

$$\frac{\partial f(x_c)}{\partial x^*} = \left.\frac{\partial f(x,x^*)}{\partial x^*}\right|_{x=Const} = \frac{1}{2}\left(\frac{\partial f}{\partial a} + j\,\frac{\partial f}{\partial b}\right). \tag{13}$$

As an example, let $f(x_c) = f(x,x^*) = x^*x = \|x\|^2 = a^2 + b^2$ be a real function of complex variable which is the squared Euclidean distance to the origin, with $x = a + j\,b$. Then,

$$f(x_c) = f(x,x^*) = x^*x = \underbrace{a^2 + b^2}_{=u} + j\,\underbrace{(ab - ab)}_{=v} = y \tag{14}$$

as $v = 0$; clearly the Cauchy-Riemann equations do not hold, and hence $f(x_c) = f(x,x^*) = x^*x$ is not analytic and thus is non-holomorphic function. To overcome this apparent difficult, by applying the *CR-Calculus* leads to

$$\frac{\partial f(x_c)}{\partial x} = x^*; \quad \frac{\partial f(x_c)}{\partial x^*} = x, \tag{15}$$

which suggests the geometric interpretation shown in **Figure 1**. Its analysis allows us to infer that the direction of maximum rate of change of the objective function is given by the conjugate gradient defined in Eq. (13). Observe that its positive direction is referred to a maximization problem (dot arrow), whereas the opposite direction concerns to the cost function minimization.

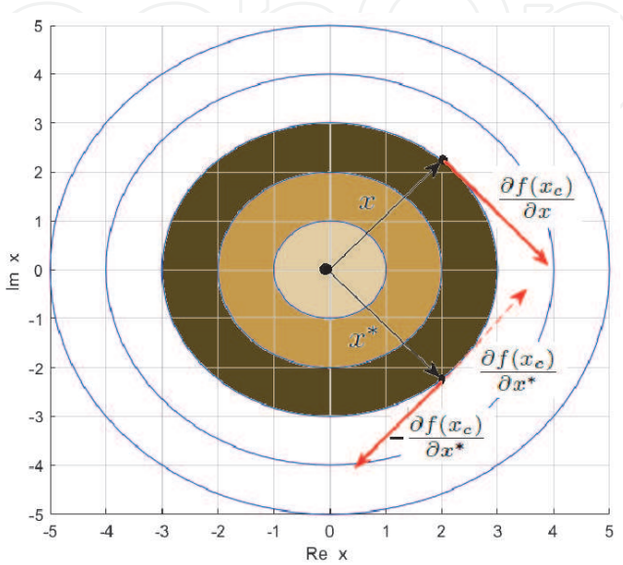

Figure 1.
Contour plot of the real function of complex variable.

Hereafter, a real-valued or complex-valued function and its argument are provided with a subscript c if it is a function in the complex conjugate coordinates, i.e., (x, x^*). Moreover, when the *CR-Calculus* is extended to the vector case, it is denoted that the multivariate *CR-Calculus* and the basic rules for the scalar case remain unchanged.

3. Solution of the problem: $\mathbf{A}_{m\times n}\,\underline{\boldsymbol{x}}_{n\times 1} = \underline{\boldsymbol{y}}_{m\times 1} \in \mathbb{C}^{m\times n}$

As well in the real domain, there are two classes of methods for the numerical solution of large linear system of equations in complex plane:

- Direct methods: Which produce the exact solution assuming the absence of truncation and round-off errors, by performing a finite number of *flops* in a finite known number of steps. These methods are usually recommended when most of the entries in the coefficient matrix are nonzero and the dimension of the system is not too large, for instance, the Gaussian elimination, the LU decomposition and QR factorization, to cite a few.

- Iterative methods: This class of methods is beyond of this work. Notice the solution provided by these methods is approximated and the accuracy is imposed by the user. The number of ν accomplished iterations depends on the given precision or convergence criterion. This class of methods is preferred when the majority of the coefficients are equal to zero and the number of unknowns is very large. The methods of Jacobi and Gauss-Seidel are good examples, besides the classical Conjugate gradient method [5].

3.1 Three-angle complex rotation algorithm

The first studied algorithm is the *three-angle complex rotations (TACR)*, which is derived in polar coordinates [6]. Nonetheless, the key idea behind *QR* decomposition is to eliminate the square roots needed for the computation of the cosine and sine which represent a bottleneck in real-time applications. Consequently, this algorithm is devoid of interest for the solution of large linear systems of equations.

3.2 Complex-valued fast givens rotations

On the other hand, the *fast plane rotation* [7] that is derived in complex plane and rectangular coordinates is a square root- and division-free Givens rotations [8]. In this sense, the fast plane rotation which is also referred as the complex-valued fast Givens rotations ($CV-FGR$) is a very efficient algorithm, aiming a *QR-decomposition* of matrices once the computations are performed incrementally, i.e., as the data arrives sequentially in time. Thus, it allows us to reduce the overall latency and hardware resources drastically. In the forthcoming contribution, the $CV-FGR$ performance will be compared to the well-known approaches which were successfully applied to the power system state estimation [9–11], once they are accordingly converted from real to complex domain. Further proposals in the updated state of the art will be equally considered, e.g., [12] and [13], to cite a few.

The complex fast Givens transformation $\mathbf{M}$ is computed using **Algorithm 1** such that the second component of $\mathbf{M}^H\,\underline{x}$ is zero and $\mathbf{M}^H\mathbf{D}\mathbf{M}$ is a diagonal matrix, as shown below:

$$\mathbf{M} = \underbrace{\begin{bmatrix} \beta & 1 \\ 1 & \alpha \end{bmatrix}}_{type=1} \quad or \quad \underbrace{\begin{bmatrix} 1 & \alpha \\ \beta & 1 \end{bmatrix}}_{type=2} \tag{16}$$

$$\mathbf{M}^H \underline{x} = \begin{bmatrix} r \\ 0 \end{bmatrix} \quad \& \quad \mathbf{M}^H \underbrace{\begin{bmatrix} d_f & 0 \\ 0 & d_g \end{bmatrix}}_{=\mathbf{D}} \mathbf{M} = \underbrace{\begin{bmatrix} d_f^{new} & 0 \\ 0 & d_g^{new} \end{bmatrix}}_{=\mathbf{D}^{\mathbf{new}}}. \tag{17}$$

Notice the superscript H denotes the *Hermitian* operation, i.e., complex conjugate transpose.

Algorithm 1. Complex fast Givens transform.

$[\alpha, \beta, \mathrm{r}, type, d_f^{new}, d_g^{new}]$ = **fast.givens** (f, g, d_f, d_g);
if $f = 0$ **then**
  $type = 1; \alpha = \beta = 0; r = g;$
  $d_f^{new} = d_g;\ d_g^{new} = d_f;$
else if $g = 0$ **then**
  $type = 2; \alpha = \beta = 0; r = f;$
  $d_f^{new} = d_f;\ d_g^{new} = d_g;$
else if $\|f\|^2 \le \|g\|^2$ **then**
  $type = 1; i = f/g; s = d_g/d_f;$
  $\alpha = -i; \beta = s * i;$
  $\gamma = s * \|i\|^2; r = g * (1+\gamma);$
  $d_f^{new} = (1+\gamma) * d_g;\ d_g^{new} = (1+\gamma) * d_f;$
else
  $type = 2; i = g/f; s = d_f/d_g;$
  $\alpha = -i; \beta = s * i;$
  $\gamma = s * \|i\|^2; r = f * (1+\gamma);$
  $d_f^{new} = (1+\gamma) * d_f;\ d_g^{new} = (1+\gamma) * d_g;$
end if

The matrices **Q**, **M**, and **D** are connected to the following equation:

$$\mathbf{Q} = \mathbf{M}\left(\mathbf{D}^{\mathbf{new}}\right)^{-1/2} = \mathbf{M}\, diag\left(1/sqrt\left(d_i^{new}\right)\right). \tag{18}$$

In the sequence the *QR-decomposition* using complex fast Givens transformations is presented as **Algorithm 2**.

Algorithm 2. Sequential fast Givens QRD decomposition.

$[Q_\alpha, Q_\beta, type, R]$=**fast.givens_QRD** (A);
[m,n]=size(A);
$R = zeros(n); D = I_n;$
$R(1, 1:n) = A(1, 1:n);$
for $i = 2 : m$ **do**
  new$= A(i, 1:n); d_{new} = 1;$
  if $i <= n$ **then**
    $k = i;$
  else

$k = n + 1$;
end if
for $j = 1 : k - 1$ **do**
Step 1: Get *alpha* **and** *beta* **using Algorithm 1:**
$[\alpha, \beta, \mathbf{R}(j,j), type, d_f^{new}, d_g^{new}]$=**fast.givens** ($\mathbf{R}(j,j)$, **new**$(1,j)$, d_f, d_g);
Step 2: Update elements based on *type*
if $j < n$ and *type*=1 **then**

$$\begin{bmatrix} R(j, j+1:n) \\ new(1, j+1:n) \end{bmatrix} = \begin{bmatrix} 1 & \beta \\ \alpha & 1 \end{bmatrix}^H \begin{bmatrix} R(j, j+1:n) \\ new(1, j+1:n) \end{bmatrix}$$

else if $J < n$ and *type*=2 **then**

$$\begin{bmatrix} R(j, j+1:n) \\ new(1, j+1:n) \end{bmatrix} = \begin{bmatrix} \beta & 1 \\ 1 & \alpha \end{bmatrix}^H \begin{bmatrix} R(j, j+1:n) \\ new(1, j+1:n) \end{bmatrix}$$

end if
end for
if $k < = n$ **then**
$R(k, 1:n) = new(1, 1:n); d(k) = d_{new}$;
end if
end for

Note that the complex fast Givens QRD does not require any square root operation, and during each incremental QRD-update step, the incoming input data row-vector is stored, i.e., vector **new**. In the sequence, the input data row-vector elements are zero-out (inner for loop) in order to update upper triangular matrix **R**. The **new** vector is overwritten each time till the *QRD*-algorithm has exhausted all the input data, i.e., the upper triangular matrix **R** is entirely updated.

4. Complex-valued Newton-Raphson method

Aiming the solution of any set of exactly determined equations in complex plane, the vector of unknowns is regularly taken into account in the iterative algorithm as follows:

$$\underline{x}_c = \left[x_1, x_2, \ldots, x_{N-1}, x_1^*, x_2^*, \ldots, x_{N-1}^*\right]^T, \tag{19}$$

and the residual vector hereafter termed as "mismatches" vector leads to

$$\underline{M}(\underline{x}_c) = \left[M_1, M_2, \ldots, M_{N-1}, M_1^*, M_2^*, \ldots, M_{N-1}^*\right]^T. \tag{20}$$

Nonetheless, here the goal is to calculate $\underline{x}_c$ that satisfies

$$\underline{M}(\underline{x}_c) = \underline{Y_e}(\underline{x}_c) - \underline{Y_s} = 0, \tag{21}$$

where in Eq. (21), $\underline{Y_s}$ is a vector of specified quantities, i.e., constant term; $\underline{Y_e}(\underline{x}_c) = \mathbf{J}_c\ \Delta\underline{x}_c$ is a vector of calculated quantities at each iteration. Consequently, the linearization of Eq. (21) from one step to the sequel leads to

$$\underline{M}\left(\underline{x}_c^{(\nu-1)}\right) + \mathbf{J}^{(\nu-1)}\ \Delta\underline{x}_c^{(\nu)} = 0, \tag{22}$$

or

$$\Delta \underline{x}_c^{(\nu)} = -\left(\mathbf{J}^{(\nu-1)}\right)^{-1} \underline{M}\left(\underline{x}_c^{(\nu-1)}\right), \tag{23}$$

where **J** is the complex-valued Jacobian matrix which the dimension is $2\,(N-1) \times 2\,(N-1)$. It means that at least one complex-valued state variable have to be specified, i.e., is known.

As a further advantage provided by the Wirtinger calculus [2, 3], the Jacobian matrix which emerged in Cartesian coordinates needs lesser algebra task as well as minor implementation effort (encoding) than the former procedure in real domain [14]. Thereby, the Jacobian matrix in expanded form may be represented through four partitions matrix, yielding

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \underline{M}_1}{\partial \underline{x}_1} & \frac{\partial \underline{M}_1}{\partial \underline{x}_2} & \cdots & \frac{\partial \underline{M}_1}{\partial \underline{x}_{N-1}} & \vdots & \frac{\partial \underline{M}_1}{\partial \underline{x}_1^*} & \frac{\partial \underline{M}_1}{\partial \underline{x}_2^*} & \cdots & \frac{\partial \underline{M}_1}{\partial \underline{x}_{N-1}^*} \\ \frac{\partial \underline{M}_2}{\partial \underline{x}_1} & \frac{\partial \underline{M}_2}{\partial \underline{x}_2} & \cdots & \frac{\partial \underline{M}_2}{\partial \underline{x}_{N-1}} & \vdots & \frac{\partial \underline{M}_2}{\partial \underline{x}_1^*} & \frac{\partial \underline{M}_2}{\partial \underline{x}_2^*} & \cdots & \frac{\partial \underline{M}_2}{\partial \underline{x}_{N-1}^*} \\ \frac{\partial \underline{M}_{N-1}}{\partial \underline{x}_1} & \frac{\partial \underline{M}_{N-1}}{\partial \underline{x}_2} & \cdots & \frac{\partial \underline{M}_{N-1}}{\partial \underline{x}_{N-1}} & \vdots & \frac{\partial \underline{M}_{N-1}}{\partial \underline{x}_1^*} & \frac{\partial \underline{M}_{N-1}}{\partial \underline{x}_2^*} & \cdots & \frac{\partial \underline{M}_{N-1}}{\partial \underline{x}_{N-1}^*} \\ \cdots & \cdots & \cdots\cdots & \cdots & \vdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial \underline{M}_1^*}{\partial \underline{x}_1} & \frac{\partial \underline{M}_1^*}{\partial \underline{x}_2} & \cdots & \frac{\partial \underline{M}_1^*}{\partial \underline{x}_{N-1}} & \vdots & \frac{\partial \underline{M}_1^*}{\partial \underline{x}_1^*} & \frac{\partial \underline{M}_1^*}{\partial \underline{x}_2^*} & \cdots & \frac{\partial \underline{M}_1^*}{\partial \underline{x}_{N-1}^*} \\ \frac{\partial \underline{M}_2^*}{\partial \underline{x}_1} & \frac{\partial \underline{M}_2^*}{\partial \underline{x}_2} & \cdots & \frac{\partial \underline{M}_2^*}{\partial \underline{x}_{N-1}} & \vdots & \frac{\partial \underline{M}_2^*}{\partial \underline{x}_1^*} & \frac{\partial \underline{M}_2^*}{\partial \underline{x}_2^*} & \cdots & \frac{\partial \underline{M}_2^*}{\partial \underline{x}_{N-1}^*} \\ \frac{\partial \underline{M}_{N-1}^*}{\partial \underline{x}_1} & \frac{\partial \underline{M}_{N-1}^*}{\partial \underline{x}_2} & \cdots & \frac{\partial \underline{M}_{N-1}^*}{\partial \underline{x}_N} & \vdots & \frac{\partial \underline{M}_{N-1}^*}{\partial \underline{x}_1^*} & \frac{\partial \underline{M}_{N-1}^*}{\partial \underline{x}_2^*} & \cdots & \frac{\partial \underline{M}_{N-1}^*}{\partial \underline{x}_{N-1}^*} \end{bmatrix}. \tag{24}$$

For instance, **Figure 2** displays the pattern for the IEEE-57 bus system in $\mathbb{R}$-domain and complex plane. This latter is given by Eq. (24).

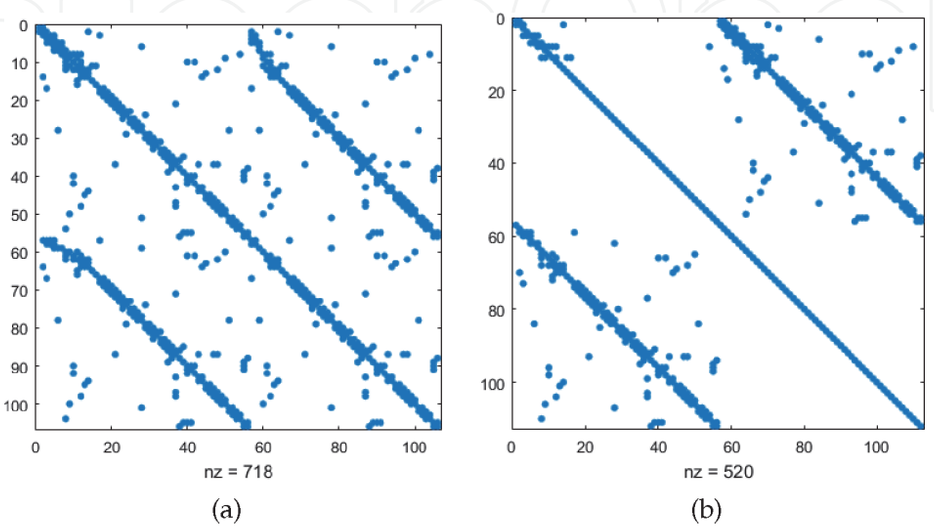

Figure 2.
Sparsity structure of (a) real-valued Jacobian matrix; (b) complex-valued Jacobian matrix of the IEEE 57-bus system.

4.1 Jacobian matrix factorization

In order to factorize the Jacobian matrix required in Eq. (23), the recommended procedure is to operate the factorization on the augmented Jacobian matrix, yielding

$$\mathbf{J}_a \quad = \left[\mathbf{J} \ \underline{M}\left(\underline{x}_c^{(\nu-1)}\right)\right], \tag{25}$$

which the dimension is $2n \times (2n+1)$, resulting

$$\tilde{\mathbf{J}}_a \quad = \left[\mathbf{T}_c \ \tilde{\underline{M}}\left(\underline{x}_c^{(\nu-1)}\right)\right], \tag{26}$$

where $\mathbf{T_c}$ is an upper triangular matrix of dimension $(2n \times 2n)$ and $\tilde{\underline{M}}\left(\underline{x}_c^{(\nu-1)}\right)$ comprises the corresponding rows in the updated *rhs* vector of dimension $(2n \times 1)$. Finally, Eq. (23) is solved by performing a back-substitution via

$$\Delta\underline{x}_c^{(\nu)} = \mathbf{T_c}\ \tilde{\underline{M}}\left(\underline{x}_c^{(\nu-1)}\right). \tag{27}$$

On the other hand, it is recommendable to perform the convergence checking over the infinity norm of two vectors. Firstly, as the former, it occurs over the corrections to be applied to the state variables and simultaneously over the mismatches vector. This latter is included, aiming to be aware against ill-conditioned systems [14], yielding

$$\left\|\Delta\underline{x}_c^{(\nu)}\right\|_\infty \ and \ \left\|M(\underline{x}_c)^{(\nu)}\right\|_\infty \leq tol \ \left(e.g., 10^{-12}\right). \tag{28}$$

If Eq. (28) is satisfied, stop and print out the results. Otherwise, the state vector is updated as shown below:

$$\underline{x}_c^{(\nu)} = \underline{x}_c^{(\nu-1)} + \Delta\underline{x}_c^{(\nu)}, \tag{29}$$

and the iteration counter is increased followed by the updating of the mismatch vector and the Jacobian matrix factorization. This latter task can be mandatory or not once the Jacobian matrix may be kept constant throughout the iterative process (approximate, instead of full gain) which is a decision very often adopted after the second iteration aiming to lighten the computational burden. Further details can be found in [14], but in the sequence, a small example is presented forwarded of simulations carried out on large systems. However, as any other application in the industry, the power flow model lies in the solution of a system of linear algebraic equations which is summarized thereafter.

4.2 Nodal equation

This approach requires the nodal admittance matrix building, e.g.,

$$\underline{I} = \mathbf{Y_{bus}}\ \underline{V}, \tag{30}$$

thus the complex nodal power can be expressed as

$$\underline{S} = diag(\underline{V})\ \underline{I}^*, \tag{31}$$

or

$$\underline{S} = diag(\underline{V})\ \mathbf{Y}_{bus}^{*}\ \underline{V}^{*}. \tag{32}$$

Then, the nodal complex power at $bus - k$, i.e., S_k, is

$$S_k = V_k\, y_{kk}^{*}\ V_k^{*} + V_k \sum_{\substack{m=0 \\ m\neq k}}^{N} y_{km}^{*}\ V_m^{*}, \tag{33}$$

where N is the number of network nodes. Therefore, the unknowns to be determined are the voltages at each node or bus into the system and the general power flow equations that model any type of branch in an electrical network, i.e., transmission lines and phase- and phase-shifting-transformers can be written, yielding

$$S_{km} = V_k \left(\frac{y_{km}^{*}}{t_{km} t_{km}^{*}} - j\, b_{km}^{sh} \right) V_k^{*} - V_k\, \frac{y_{km}^{*}}{t_{km}}\ V_m^{*}, \tag{34}$$

$$S_{mk} = V_m \left(y_{km}^{*} - j\, b_{km}^{sh} \right) V_m^{*} - V_m\, \frac{y_{km}^{*}}{t_{km}^{*}}\ V_k^{*}. \tag{35}$$

and their complex conjugate counterpart are

$$S_{km}^{*} = V_k^{*} \left(\frac{y_{km}}{t_{km}^{*} t_{km}} + j\, b_{km}^{sh} \right) V_k - V_k^{*}\, \frac{y_{km}}{t_{km}^{*}}\ V_m, \tag{36}$$

$$S_{mk}^{*} = V_m^{*} \left(y_{km} + j\, b_{km}^{sh} \right) V_m - V_m^{*}\, \frac{y_{km}}{t_{km}}\ V_k. \tag{37}$$

In Eqs. (34)–(37), $t_{km} = a_{km}\, e^{-j\varphi_{km}}$ is the general off-nominal tap transformer model which is composed by an ideal transformer with complex turns ratio $t_{km} : 1$ in series with its admittance or impedance. Thus, if the corresponding branch is referred to.

1. Off-nominal tap transformer: $b_{km}^{sh} = 0$ and $\varphi_{km} = 0$.
2. Pure-shifter: $b_{km}^{sh} = 0$ and $a_{km} = 1$.
3. Phase-shifter: $b_{km}^{sh} = 0$.
4. $\pi-$transmission line: $a_{km} = 1$ and $\varphi_{km} = 0$.

4.3 Small example

The power flow model in complex plane as detailed in [14] is applied to a small example system in which the diagram is shown in **Figure 3**, while the corresponding branch parameters and bus data, both in pu ($V_{base} = 230\, kV$; $S_{base} = 100\, MVA$), are presented in **Table 1**.

The nodal admittance matrix, $\mathbf{Y}_{bus}$, leads to

$$\mathbf{Y}_{bus} = \begin{bmatrix} +213.3474 - j\,380.8922 & -205.1282 + j\,358.9744 & -8.2192 + j\,21.9178 \\ -205.1282 + j\,358.9744 & +205.2414 - j\,359.3821 & -0.1132 + j\,0.6037 \\ -8.2192 + j\,21.9178 & -0.1132 + j\,0.6037 & +8.3324 - j\,22.3256 \end{bmatrix}. \tag{38}$$

The whole set of intermediary results throughout the iterative process is presented in the sequence.

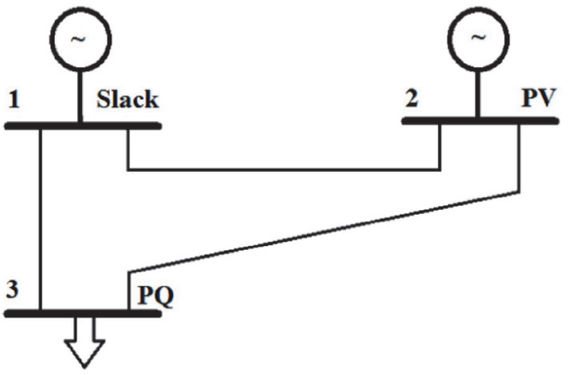

Figure 3.
Small 3-bus system.

Branch	**Series**		**Shunt**	
$i \rightarrow j$	**R**	**X**	**Charging**	**Y/2**
	pu	**pu**	**MVAr**	**pu**
1-2	0.0012	0.0021	39.2	0.196
1-3	0.0150	0.0400		
2-3	0.3000	1.6000		
Bus	**Specified quantities in pu**			
Type	P_g	V	P_{load}	Q_{load}
PV-2	1.0000	1.0000	0.2160	0.0918
PQ-3			2.700	1.620

Table 1.
Branch and bus data.

As in the real domain, the elements of the complex-valued Jacobian matrix remain practically unchanged after the second iteration, which suggest that we may keep them constant thereafter. Moreover, the computation of some entries can be avoided because they are complex conjugates of other entries; it turns out that these elements are PV-nodes.

$$\mathbf{J}^{(\nu=0)} = \left[\begin{array}{cc|cc} 413.6193 \times e^{-j\,60.268} & 0.4232 \times e^{-j\,74.484} & 413.6193 \times e^{+j\,60.268} & 0.4232 \times e^{+j\,74.484} \\ & 0.1960 \times e^{+j\,90.000} & 0.6143 \times e^{-j\,100.619} & 23.8298 \times e^{+j\,69.533} \\ 1.0000 \times e^{j\,0.000} & & 1.0000 \times e^{j\,0.000} & \\ 0.6143 \times e^{+j\,100.619} & 23.8298 \times e^{-j\,69.533} & & 0.1960 \times e^{+j\,90.000} \end{array}\right],$$

$$\mathbf{J}^{(\nu=1)} = \left[\begin{array}{cc|cc} 414.4312 \times e^{-j\,60.323} & 0.4232 \times e^{-j\,74.616} & 414.4312 \times e^{+j\,60.323} & 0.3842 \times e^{+j\,69.158} \\ & 3.1443 \times e^{-j\,148.587} & 0.5577 \times e^{-j\,105.946} & 21.6334 \times e^{+j\,64.207} \\ 1.0000 \times e^{-j\,0.132} & & 1.0000 \times e^{+j\,0.132} & \\ 0.5577 \times e^{+j\,105.946} & 23.8298 \times e^{-j\,69.665} & & 3.1443 \times e^{+j\,148.587} \end{array}\right],$$

$$\mathbf{J}^{(\nu=2)} = \left[\begin{array}{cc|cc} 414.3163 \times e^{-j\,60.313} & 0.4232 \times e^{-j\,74.599} & 414.3180 \times e^{+j\,60.313} & 0.3765 \times e^{+j\,68.935} \\ & 3.5321 \times e^{-j\,143.603} & 0.5465 \times e^{-j\,106.168} & 21.2006 \times e^{+j\,63.984} \\ 1.0000 \times e^{-j\,0.115} & & 1.0000 \times e^{+j\,0.115} & \\ 0.5452 \times e^{+j\,106.041} & 23.8298 \times e^{-j\,69.648} & & 3.5232 \times e^{+j\,144.712} \end{array}\right],$$

$$\mathbf{J}^{(\nu=3)} = \left[\begin{array}{cc|cc} 414.3115 \times e^{-j\,60.313} & 0.4232 \times e^{-j\,74.599} & 414.3133 \times^{+j\,60.313} & 0.3763 \times e^{+j\,68.932} \\ & 3.5408 \times e^{-j\,143.485} & 0.5463 \times e^{-j\,106.171} & 21.1905 \times e^{+j\,63.982} \\ 1.0000 \times e^{-j\,0.115} & & 1.0000 \times e^{+j\,0.115} & \\ 0.5449 \times e^{+j\,106.040} & 23.8298 \times e^{-j\,69.648} & & 3.5316 \times e^{+j\,144.620} \end{array}\right].$$

Interestingly, the numerical values of the state variables corrections, state variables, and mismatches vectors calculated in the complex plane are displayed in the **Tables 2–4**, respectively.

4.4 Performance in larger systems

The algorithm described earlier was encoded in MATLAB by using sparsity technique and column approximate minimum degree (*colamd*) ordering scheme. The numerical tests were executed by using an Intel® Core™ i5-4200 CPU at 1.60 Hz 2.30 GHz, 6GB of RAM, and 64-bit operating system. A flat start condition is assigned to the state variables in all simulations.

Thereby, **Table 5** presents the performance on larger systems of the Newton-Raphson method in complex plane which is highlighted in bold. The corresponding performance is compared with those of the former Newton-Raphson method developed in polar and rectangular coordinates, both in real domain. In all simulations the convergence criterion of 1×10^{-12} is assumed.

$\underline{\Delta x}$	$\underline{\Delta x}^{(\nu=0)}$	$\underline{\Delta x}^{(\nu=1)}$	$\underline{\Delta x}^{(\nu=2)}$	$\underline{\Delta x}^{(\nu=3)}$
Δx_2	$0.0023 \times e^{+j\ 90.00}$	$0.0003 \times e^{-j\ 90.39}$	$0.0127 \times e^{-j\ 90.08}$	$0.1708 \times e^{+j\ 89.92}$
Δx_3	$0.1278 \times e^{-j\ 138.75}$	$0.0185 \times e^{-j\ 174.56}$	$0.4267 \times e^{-j\ 179.44}$	$0.2326 \times e^{-j\ 179.18}$
Δx_2^*	$0.0023 \times e^{-j\ 90.00}$	$0.0003 \times e^{+j\ 90.37}$	$0.0127 \times e^{+j\ 90.07}$	$0.1708 \times e^{-j\ 89.91}$
Δx_3^*	$0.1278 \times e^{+j\ 138.75}$	$0.0203 \times e^{+j\ 178.81}$	$0.4795 \times e^{+j\ 173.22}$	$0.2615 \times e^{+j\ 173.47}$
$\|\Delta \underline{x}\|_\infty$ [a]	0.127809	0.020316	4.795255×10^{-4}	2.614490×10^{-7}

[a] *Convergence criteria:* $\|\Delta \underline{X}\|_\infty < tol. \approx 10^{-4}$.

Table 2.
Unknown correction vector.

$\underline{x}$	$\underline{x}^{(\nu=0)}$	$\underline{x}^{(\nu=1)}$	$\underline{x}^{(\nu=2)}$	$\underline{x}^{(\nu=3)}$
x_2	$1.000 \times e^{j\ 0.0}$	$1.000 \times e^{+j\ 0.132}$	$1.000 \times e^{+j\ 0.115}$	$1.000 \times e^{+j\ 0.115}$
x_3	$1.000 \times e^{j\ 0.0}$	$0.907 \times e^{-j\ 5.326}$	$0.889 \times e^{-j\ 5.548}$	$0.889 \times e^{-j\ 5.552}$
x_2^*	$1.000 \times e^{-j\ 180}$	$1.000 \times e^{-j\ 0.132}$	$1.000 \times e^{-j\ 0.115}$	$1.000 \times e^{-j\ 0.115}$
x_3^*	$1.000 \times e^{-j\ 180}$	$0.907 \times e^{+j\ 5.326}$	$0.887 \times e^{+j\ 5.421}$	$0.887 \times e^{+j\ 5.420}$

Table 3.
Unknown vector.

$\underline{M}$	$\underline{M}^{(\nu=0)}$	$\underline{M}^{(\nu=1)}$	$\underline{M}^{(\nu=2)}$	$\underline{M}^{(\nu=3)} \times \mathbf{10^{-3}}$
M_2	$-1.5680 + j\ 0.0000$	$+0.2178 + j\ 0.0000$	$+0.0093 + j\ 0.0000$	$+0.1229 + j\ 0.0000$
M_3	$+2.7000 + j\ 1.4240$	$+0.1363 + j\ 0.3648$	$+0.0021 + j\ 0.0087$	$+0.0011 + j\ 0.0047$
M_2^*	$+0.0000 + j\ 0.0000$	$+0.0000 - j\ 0.0000$	$+0.0000 - j\ 0.0000$	$+0.0000 - j\ 0.0000$
M_3^*	$+2.7000 - j\ 1.4240$	$+0.1363 - j\ 0.3648$	$0.0021 - j\ 0.0087$	$0.0011 - j\ 0.0047$

Convergence criteria: $\|\underline{M}\|_\infty < tol. \approx 10^{-4}$.

Table 4.
Mismatch vector.

	CV-Jacobian matrix dimension $2(N-1) \times 2(N-1)$	Algorithms	Number of iterations	Time/iteration (s)	Total time (s)
IEEE-14		$1.RV-NRM^{(p)}$	5	0.0184	0.1647
		$2.RV-NRM^{(r)}$	5	0.0150	0.0942
	26×26	**$3.\mathbf{CV-NRM^{(r)}}$**	**5**	**0.0098**	**0.0767**
IEEE-30		$1.RV-NRM^{(p)}$	5	0.0206	0.1791
		$2.RV-NRM^{(r)}$	6	0.0105	0.1121
	58×58	**$3.\mathbf{CV-NRM^{(r)}}$**	**6**	**0.0091**	**0.0645**
IEEE-57		$1.RV-NRM^{(p)}$	6	0.0132	0.1653
		$2.RV-NRM^{(r)}$	6	0.0137	0.1303
	112×112	**$3.\mathbf{CV-NRM^{(r)}}$**	**6**	**0.0110**	**0.0810**
IEEE-118		$1.RV-NRM^{(p)}$	5	0.0162	0.1575
		$2.RV-NRM^{(r)}$	6	0.0196	0.1840
	234×234	**$3.\mathbf{CV-NRM^{(r)}}$**	**5**	**0.0121**	**0.1056**
SIN-1916		$1.RV-NRM^{(p)}$	7	0.4642	3.4732
		$2.RV-NRM^{(r)}$	8	0.3095	2.6430
	3830×3830	**$3.\mathbf{CV-NRM^{(r)}}$**	**7**	**0.7699**	**4.5561**

(p)—polar coordinates; (r)—rectangular coordinates; tol., 1×10^{-12}.

Table 5.
Performance in larger systems—squared matrices.

The results presented in the aforementioned table allows us to infer that the Newton-Raphson method in complex plane has very good performance. Except for the SIN-1916 bus system, the time consuming required to achieve the solution is lower than the remainder approaches.

In the next section, the Gauss-Newton method is presented. The goal is to solve overdetermined systems of equations, i.e., the former nonlinear least-squares method in complex plane.

5. Complex-valued weighted-least-squares method (CV-WLS)

As shown in [3], the complex-valued WLS state estimator minimizes an objective function defined as

$$\underset{\underline{x}c}{\operatorname{argmin}}\ \mathcal{J}\left(\underline{x}_c\right) = \frac{1}{2}\left(\underline{z}_c - \underline{h}_c(\underline{x}_c)\right)^H \Omega_c^{-1}\left(\underline{z}_c - \underline{h}_c(\underline{x}_c)\right), \tag{39}$$

where $\underline{z}_c = (\underline{z}, \underline{z}^*)$ is a vector of specified complex values of dimension $(2m \times 1)$, $\underline{x}_c = (\underline{x}, \underline{x}^*)$ is a vector of complex-valued state variables of dimension $(2n \times 1)$, $\underline{h}_c(\underline{x}_c)$ is a vector of nonlinear functions of dimension $(2m \times 1)$ that maps $\underline{z}_c$ to $\underline{x}_c$, $\underline{\omega}_c$ is a vector of random errors in complex plane which dimension is $(2m \times 1)$, and

$\Omega_c = E\left(\underline{\omega}_c\ \underline{\omega}_c^H\right)$ is a Hermitian positive-definite covariance matrix of $\underline{\omega}_c$ which dimension is $(2m \times 2m)$. The superscript $(\cdot)^H$ stands for Hermitian operator, i.e., the transpose complex conjugate operation. Thus, the necessary condition of optimality is given by

$$\frac{\partial \mathcal{J}\ (\underline{x}_c)}{\partial \underline{x}_c} = -\mathbf{H}(\underline{x}_c)^H\ \Omega_c^{-1}\ (\underline{z}_c - \underline{h}_c(\underline{x}_c)) = 0. \tag{40}$$

By applying a first-order Taylor series expansion of $\underline{h}_c(\underline{x}_c)$ about $\underline{x}_c^{(\nu)}$, we get

$$\underline{h}_c(\underline{x}_c) = \underline{h}_c\left(\underline{x}_c^{(\nu)}\right) + \mathbf{H}\left(\underline{x}_c^{(\nu)}\right)\ \left(x_c - \underline{x}_c^{(\nu)}\right). \tag{41}$$

By replacing Eq. (41) into Eq. (40), we obtain

$$\mathbf{H}\left(\underline{x}_c^{(\nu)}\right)^H\ \Omega_c^{-1}\ \left[\underline{z}_c - \underline{h}_c\left(\underline{x}_c^{(\nu)}\right) - \mathbf{H}\left(\underline{x}_c^{(\nu)}\right)\ \left(x_c - \underline{x}_c^{(\nu)}\right)\right] = 0, \tag{42}$$

yielding the updated estimated state vector expressed as

$$\begin{aligned} \underline{x}_c^{(\nu+1)} &= \underline{x}_c^{(\nu)} + \mathbf{G}\left(\underline{\mathbf{x}}_\mathbf{c}^{(\nu)}\right)^{-1}\ \mathbf{H}\left(\underline{x}_c^{(\nu)}\right)^H\ \Omega_c^{-1}\ \Delta\underline{z}_c^{(\nu)}, \\ &= \underline{x}_c^{(\nu)} + \Delta\underline{x}_c^{(\nu)}, \end{aligned} \tag{43}$$

where

$$\Delta\underline{x}_c^{(\nu)} = \mathbf{G}\left(\underline{x}_c^{(\nu)}\right)^{-1}\ \mathbf{H}\left(\underline{\mathbf{x}}_c^{(\nu)}\right)^H\ \Omega_c^{-1}\ \Delta\underline{z}_c^{(\nu)}. \tag{44}$$

Notice $\mathbf{G}\left(\underline{x}_c^{(\nu)}\right) = \mathbf{H}\left(\underline{x}_c^{(\nu)}\right)^H\ \Omega_c^{-1}\ \mathbf{H}\left(\underline{x}_c^{(\nu)}\right)$ and $\Delta\underline{z}_c^{(\nu)} = \underline{z}_c - \underline{h}_c\left(\underline{x}_c^{(\nu)}\right)$. Thus, the iterations are stopped when

$$\left|\left|\Delta\underline{x}_c^{(\nu)}\right|\right|_\infty \leq tol, e.g., 10^{-3}, \tag{45}$$

where $\|\cdot\|_\infty$ is the infinity norm and ν is the iteration counter.

Note that in Eq. (40), $\mathbf{H}(\underline{x}_c)$ is the Jacobian matrix of dimension $(2m \times 2n)$ defined in the complex domain, yielding

$$\mathbf{H}(\underline{x}_c) \triangleq \frac{\partial \underline{h}_c(\underline{x}_c)}{\partial \underline{x}_c} \triangleq \begin{pmatrix} \dfrac{\partial \underline{h}_c(\underline{x}_c)}{\partial \underline{x}} & \dfrac{\partial \underline{h}_c(\underline{x}_c)}{\partial \underline{x}^*} \\ \dfrac{\partial \underline{h}_c^*(\underline{x}_c)}{\partial \underline{x}} & \dfrac{\partial \underline{h}_c^*(\underline{x}_c)}{\partial \underline{x}^*} \end{pmatrix}. \tag{46}$$

Let $\mathbf{J_h} = \frac{\partial \underline{h}_c(\underline{x}_c)}{\partial \underline{x}}$ and $\mathbf{J_h^d} = \frac{\partial \underline{h}_c(\underline{x}_c)}{\partial \underline{x}^*}$ be Jacobian submatrices of dimension $(m \times n)$. They are obtained through the Wirtinger partial derivatives with respect to the complex and the complex conjugate state variables using the chain differentiation rule. Now, let us define the Jacobian matrix as

$$\mathbf{J}_c(\underline{x}_c) = \left(\mathbf{J_h}\ \ \mathbf{J_h^d}\right). \tag{47}$$

In the important special case given by Eq. (39) where $\mathcal{J}(\underline{x}_c)$ is a real-valued function of complex variables, the following property holds

$$\mathcal{J}(\underline{x}_c) \in \mathbb{R} \Rightarrow \frac{\partial \underline{h}_c^*(\underline{x}_c)}{\partial \underline{x}} = \left(\frac{\partial \underline{h}_c(\underline{x}_c)}{\partial \underline{x}^*}\right)^* = \left(\mathbf{J}_h^{\mathbf{d}}\right)^*. \tag{48}$$

Therefore, taking into account the chain rule differentiation mentioned earlier and the property stated in Eq. (48), Eq. (46) becomes

$$\mathbf{H}(\underline{x}_c) = \begin{pmatrix} \mathbf{J_h} & \mathbf{J_h^d} \\ \left(\mathbf{J_h^d}\right)^* & \left(\mathbf{J_h}\right)^* \end{pmatrix} = \begin{pmatrix} \mathbf{J}_c(\underline{x}_c) \\ \mathbf{J}_c^*(\underline{x}_c)\ \mathbf{S} \end{pmatrix}, \tag{49}$$

where $\mathbf{S}$ is a swap operator that permutes blocks of m rows or blocks of n columns depending upon whether $\mathbf{S}$ pre-multiples or post-multiples a matrix, respectively. Moreover, this operator is an isomorphism from $\mathbb{C}$ to the dual space $\mathbb{C}^*$, which obeys the properties $\mathbf{S}^{-1} = \mathbf{S}^T = \mathbf{S}$. It shows that $\mathbf{S}$ is symmetric and is equal to its own inverse, that is, $\mathbf{S}^2 = I$. For instance, as shown in [2], this matrix is defined as

$$\mathbf{S} \triangleq \begin{bmatrix} 0 & \mathbf{I}_n \\ \mathbf{I}_n & 0 \end{bmatrix}, \tag{50}$$

where $\mathbf{I}_n$ is the $(n \times n)$-identity matrix.

Now, the complex-valued gain matrix $\mathbf{G}(\underline{\hat{x}}_c)$ in expanded form can be expressed as

$$\mathbf{G}(\underline{\hat{x}}_c) = \begin{pmatrix} \mathbf{G}_{\underline{x}\,\underline{x}} & \mathbf{G}_{\underline{x}^*\,\underline{x}} \\ \mathbf{G}_{\underline{x}\,\underline{x}^*} & \mathbf{G}_{\underline{x}^*\,\underline{x}^*} \end{pmatrix}, \tag{51}$$

where $\mathbf{G}_{\underline{x}^*\,\underline{x}^*} = \left(\mathbf{G}_{\underline{x}\,\underline{x}}\right)^*$ and $\mathbf{G}_{\underline{x}^*\,\underline{x}} = \left(\mathbf{G}_{\underline{x}\,\underline{x}^*}\right)^*$, all of dimension $(n \times n)$. Then, it follows from Eq. (47) that

$$\begin{aligned} \mathbf{G}_{\underline{x}\,\underline{x}} &= \frac{1}{2}\left[\left(\frac{\partial \underline{h}}{\partial \underline{\hat{x}}}\right)^H \Omega_c^{-1}\left(\frac{\partial \underline{h}}{\partial \underline{\hat{x}}}\right) + \left(\left(\frac{\partial \underline{h}}{\partial \underline{\hat{x}}^*}\right)^H \Omega_c^{-1}\left(\frac{\partial \underline{h}}{\partial \underline{\hat{x}}^*}\right)\right)^*\right], \\ &= \frac{1}{2}\left[\mathbf{J_h}^H\, \Omega_c^{-1}\, \mathbf{J_h} + \left(\mathbf{J_h^d}^H\, \Omega_c^{-1}\, \mathbf{J_h^d}\right)^*\right], \end{aligned} \tag{52}$$

Similarly, we get

$$\begin{aligned} \mathbf{G}_{\underline{\mathbf{x}}^*\,\underline{\mathbf{x}}} &= \frac{1}{2}\left[\left(\frac{\partial \underline{\mathbf{h}}}{\partial \underline{\hat{\mathbf{x}}}}\right)^H \Omega_c^{-1}\left(\frac{\partial \underline{\mathbf{h}}}{\partial \underline{\hat{\mathbf{x}}}^*}\right) + \left(\left(\frac{\partial \underline{\mathbf{h}}}{\partial \underline{\hat{\mathbf{x}}}^*}\right)^H \Omega_c^{-1}\left(\frac{\partial \underline{\mathbf{h}}}{\partial \underline{\hat{\mathbf{x}}}}\right)\right)^*\right], \\ &= \frac{1}{2}\left[\mathbf{J_h}^H\, \Omega_c^{-1}\, \mathbf{J_h^d} + \left(\mathbf{J_h^d}^H\, \Omega_c^{-1}\, \mathbf{J_h}\right)^*\right]. \end{aligned} \tag{53}$$

The investigation of the sparsity structure of the Jacobian matrix in complex plane given by Eq. (49), e.g., **Figure 4**, reveals that the Jacobian matrices in the $\mathbb{C}$-domain are sparser than its counterpart in the $\mathbb{R}$-domain [15]. **Figure 4** is referred to the Brazilian equivalent 730-bus system which the Jacobian matrix in the $\mathbb{C}$-domain has 10,396 nonzero elements, while in the $\mathbb{R}$-domain, it has 18,403 nonzero elements; it is about 45% sparser.

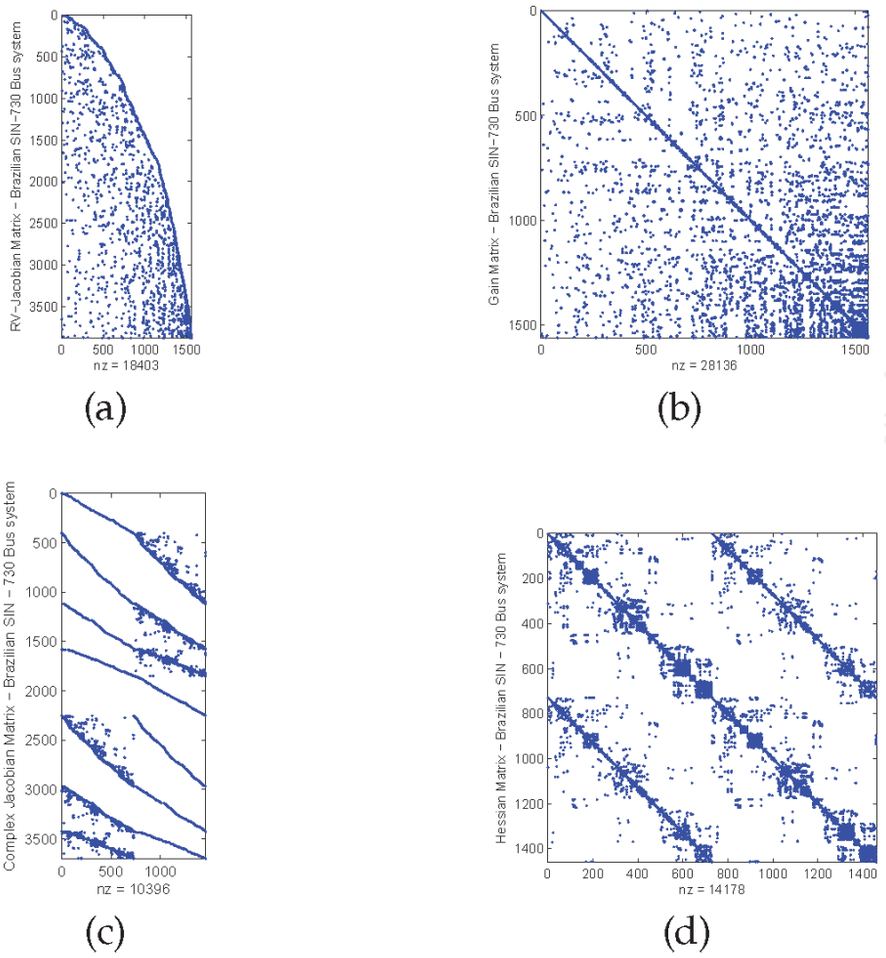

Figure 4.
Sparsity structure of (a) real-valued Jacobian matrix; (b) real-valued gain matrix; (c) complex-valued Jacobian matrix; and (d) complex-valued hessian matrix of the Brazilian equivalent 730-bus system.

Nonetheless, the recommended numerical procedure aiming to solve the complex-valued power system state estimation problem is addressed by solving the weighted form of the right-hand-side (*rhs*) of Eq. (40) instead of Eq. (44), yielding

$$\tilde{\mathbf{H}}(\underline{\hat{x}}_c)\,\Delta\underline{x}_c^{(\nu)} = \Delta\tilde{\underline{z}}_c, \tag{54}$$

where $\tilde{\mathbf{H}}(\underline{\hat{x}}_c) = \Omega_c^{-1/2}\,\mathbf{H}(\underline{\hat{x}}_c)$ is of dimension $(2m \times 2n)$ and $\Delta\tilde{\underline{z}}_c = \Omega_c^{-1/2}\,\Delta\underline{z}_c$ is of dimension $(2m \times 1)$. Thus, the incremental changes in the state vector are calculated via

$$\Delta\underline{x}_c^{(\nu)} = \tilde{\mathbf{H}}(\underline{\hat{x}}_c)^{\dagger}\,\Delta\tilde{\underline{z}}_c, \tag{55}$$

where the † operator is defined as the Moore-Penrose pseudoinverse [3].

Aiming to avoid explicitly store, the Q-matrix, we apply the QR-transformation to the augmented matrix, $\mathbf{H_a}(\underline{\hat{x}}_c)$, given by

$$\mathbf{H_a}(\underline{\hat{x}}_c) = \left[\tilde{\mathbf{H}}(\underline{\hat{x}}_c)\;\;\Delta\tilde{\underline{z}}_c\right]. \tag{56}$$

By storing the rotations in compact form, the complex-valued Jacobian matrix can be kept constant and only the right-hand-side vector is updated throughout the final iterations. The solution of the state vector increment given by Eq. (55) is found by executing a simple back-substitution of Eq. (56) after performing unitary transformations to the latter matrix, resulting in

$$\tilde{\mathbf{H}}_{\mathbf{a}}(\hat{\underline{x}}_c) = \left[\mathbf{T_c} \;\; \Delta\tilde{\tilde{\underline{z}}}_c\right]. \tag{57}$$

Here, $\mathbf{T_c}$ is an upper triangular matrix of dimension $(2n \times 2n)$, and $\Delta\tilde{\tilde{\underline{z}}}_c$ comprises the corresponding rows in the updated *rhs* vector of dimension $(2n \times 1)$. Finally, Eq. (55) is solved by performing a back-substitution via

$$\Delta\underline{x}_c^{(\nu)} = \mathbf{T_c}\, \Delta\tilde{\tilde{\underline{z}}}_c. \tag{58}$$

5.1 Small example

The one-line diagram of a 2-bus system is depicted in **Figure 5**. The system is provided with two PMU measurements that meter the nodal voltage magnitudes and phase angles and two real and reactive power flow measurements, which are identified by means of black bullets and red triangles, respectively. In **Table 6**, the transmission line parameters are given in *pu*.

From **Figure 5**, the complex power injections at sending and receiving end, respectively, are written yielding

$$S_1 = V_1 \left[\left(y_{12}^* - j\, b_{12}^{sh}\right) V_1^* - y_{12}^*\, V_2^*\right], \tag{59}$$

$$S_2 = V_2 \left[\left(y_{12}^* - j\, b_{12}^{sh}\right) V_2^* - y_{12}^*\, V_1^*\right]. \tag{60}$$

Applying the Wirtinger calculus to Eqs. (59) and (60) leads to the Jacobian matrix given by Eq. (47) at each iteration as shown in the sequence. Here, the complex power injection measurement at each end, i.e., S_1 and S_2, is equal to the corresponding power flow measurement, i.e., S_{12} and S_{21}, respectively.

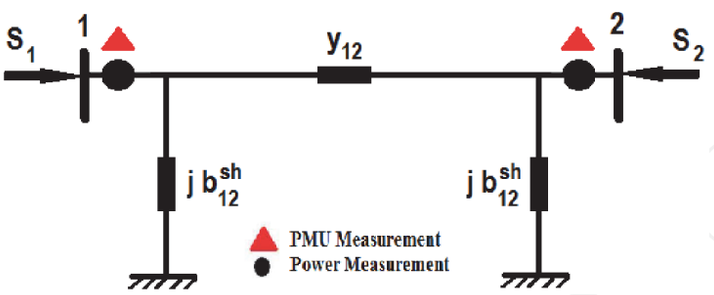

Figure 5.
2-Bus power system.

Branch	Series		Shunt	
$i \rightarrow j$	R	X	Charging	Y/2
	pu	pu	MVAr	pu
1-2	0.0203	0.1318	62.62	0.3131

Table 6.
Branch data.

$$\mathbf{J_c}^{(\nu=0)} = \left[\begin{array}{c|cc} 1.0000 + \text{j } 0.0000 & 1.0000 + \text{j } 0.0000 & \\ 0.0000 - \text{j } 0.3131 & 0.0000 + \text{j } 0.3131 & \begin{array}{cc} +1.1415 + \text{j } 7.0983 & - 1.1415 - \text{j } 7.4114 \\ - 1.1415 - \text{j } 7.4141 & +1.1415 + \text{j } 7.0983 \end{array} \\ 0.0000 - \text{j } 0.3131 & 0.0000 + \text{j } 0.3131 & \begin{array}{cc} +1.1415 + \text{j } 7.0983 & - 1.1415 - \text{j } 7.4114 \\ - 1.1415 - \text{j } 7.4141 & +1.1415 + \text{j } 7.0983 \end{array} \end{array}\right],$$

$$\mathbf{J_c}^{(\nu=1)} = \left[\begin{array}{c|cc} +1.0000 + \text{j } 0.0000 & +1.0000 + \text{j } 0.0000 & \\ +1.8414 + \text{j } 0.2179 & - 1.7683 - \text{j } 0.6884 & \begin{array}{cc} +0.9145 + \text{j } 5.6849 & - 0.9145 - \text{j } 5.9356 \\ - 2.5295 - \text{j } 4.9346 & +2.4565 + \text{j } 4.7148 \end{array} \\ +1.8414 + \text{j } 0.2179 & - 1.7683 - \text{j } 0.6884 & \begin{array}{cc} +0.9145 + \text{j } 5.6849 & - 0.9145 - \text{j } 5.9356 \\ - 2.5295 - \text{j } 4.9346 & +2.4565 + \text{j } 4.7148 \end{array} \end{array}\right],$$

$$\mathbf{J_c}^{(\nu=2)} = \left[\begin{array}{c|cc} +1.0000 + \text{j } 0.0000 & +1.0000 + \text{j } 0.0000 & \\ +1.8683 + \text{j } 0.4802 & - 1.7962 - \text{j } 1.0677 & \begin{array}{cc} +1.1653 + \text{j } 7.1651 & - 1.1658 - \text{j } 7.4813 \\ - 2.7074 - \text{j } 6.1598 & +2.6348 + \text{j } 5.8884 \end{array} \\ +1.8683 + \text{j } 0.4802 & - 1.7962 - \text{j } 1.0677 & \begin{array}{cc} +1.1653 + \text{j } 7.1651 & - 1.1658 - \text{j } 7.4813 \\ - 2.7074 - \text{j } 6.1598 & +2.6348 + \text{j } 5.8884 \end{array} \end{array}\right],$$

$$\mathbf{J_c}^{(\nu=3)} = \left[\begin{array}{c|cc} +1.0000 + \text{j } 0.0000 & +1.0000 + \text{j } 0.0000 & \\ +1.8828 + \text{j } 0.4247 & - 1.8098 - \text{j } 1.0085 & \begin{array}{cc} +1.1413 + \text{j } 7.0982 & - 1.1413 - \text{j } 7.4113 \\ - 2.7148 - \text{j } 6.1411 & +2.6417 + \text{j } 5.8704 \end{array} \\ +1.8828 + \text{j } 0.4247 & - 1.8098 - \text{j } 1.0085 & \begin{array}{cc} +1.1413 + \text{j } 7.0982 & - 1.1413 - \text{j } 7.4113 \\ - 2.7148 - \text{j } 6.1411 & +2.6417 + \text{j } 5.8704 \end{array} \end{array}\right].$$

As observed in the above expressions of $\mathbf{J_c}$, the sub-matrix $\mathbf{J_h}$ is more sparse than the sub-matrix $\mathbf{J_h^d}$, which affects the sparsity structure of the complex-valued Jacobian matrix as shown earlier in **Figure 4**. The state variables at each iteration are provided in **Table 7**, while the estimated measured quantities are given in **Table 8**.

$\underline{\boldsymbol{x}}$	$\underline{\boldsymbol{x}}^{(\nu=0)}$	$\underline{\boldsymbol{x}}^{(\nu=1)}$	$\underline{\boldsymbol{x}}^{(\nu=2)}$	$\underline{\boldsymbol{x}}^{(\nu=3)}$
V_1	1.0000	1.0097	1.0000	1.0001
	$e^{+j\ 0.0}$	$e^{-j\ 0.1015}$	$e^{+j\ 0.0018}$	$e^{+j\ 0.0004}$
V_2	1.0000	0.8973	0.8954	0.8956
	$e^{+j\ 0.0}$	$e^{-j\ 14.9708}$	$e^{-j\ 15.0924}$	$e^{-j\ 15.0904}$
V_1^*	1.0000	1.0097	1.0000	1.0001
	$e^{+j\ 0.0}$	$e^{+j\ 0.1015}$	$e^{-j\ 0.0018}$	$e^{-j\ 0.0004}$
V_2^*	1.0000	0.8973	0.8954	0.8956
	$e^{+j\ 0.0}$	$e^{+j\ 14.9708}$	$e^{+j\ 15.0924}$	$e^{+j\ 15.0904}$

Table 7.
Estimated state variables.

$\hat{\boldsymbol{z}}_i$	$\hat{\underline{\boldsymbol{z}}}^{(\nu=0)}$	$\hat{\underline{\boldsymbol{z}}}^{(\nu=1)}$	$\hat{\underline{\boldsymbol{z}}}^{(\nu=2)}$	$\hat{\underline{\boldsymbol{z}}}^{(\nu=3)}$
S_{12}	0.0000 − j 0.3131	+1.4747 + j 0.1745	+1.8873 + j 0.4815	+1.8827 + j 0.4248
S_{21}	0.0000 − j 0.3131	−1.4014 − j 0.0707	−1.8036 − j 0.5094	−1.7997 − j 0.4500
S_1	0.0000 − j 0.3131	+1.4747 + j 0.1745	+1.8873 + j 0.4815	+1.8827 + j 0.4248
S_2	0.0000 − j 0.3131	−1.4014 − j 0.0707	−1.8036 − j 0.5094	−1.7997 − j 0.4500

Table 8.
CV-estimated quantities.

$\underline{\boldsymbol{r}}^{(\nu=i)}$	$\underline{\boldsymbol{r}}^{(\nu=0)}$	$\underline{\boldsymbol{r}}^{(\nu=1)}$	$\underline{\boldsymbol{r}}^{(\nu=2)}$	$\underline{\boldsymbol{r}}^{(\nu=3)}$
r_{V_1}	$0.0000 + j\,0.0000$	$0.1991 + j\,0.0000$	$-0.0097 + j\,0.0018$	$0.0000 - j\,0.0000$
r_{V_2}	$-0.1349 - j\,0.2333$	$0.1634 - j\,0.0000$	$-0.0017 - j\,0.0015$	$0.0006 - j\,0.0001$
$r_{S_{12}}$	$1.8827 + j\,0.7375$	$0.4080 + j\,0.2499$	$-0.0046 - j\,0.0571$	$-0.0000 - j\,0.0004$
$r_{S_{21}}$	$-1.7998 - j\,0.1366$	$-0.3984 - j\,0.3790$	$0.0038 + j\,0.0597$	$-0.0001 + j\,0.0003$
r_{S_1}	$1.8827 + j\,0.7375$	$0.4080 + j\,0.2499$	$-0.0046 - j\,0.0571$	$-0.0000 - j\,0.0004$
r_{S_2}	$-1.7998 - j\,0.1366$	$-0.3984 - j\,0.3790$	$0.0038 + j\,0.0597$	$-0.0001 + j\,0.0003$
$\mathcal{J}(\underline{\hat{x}})^{(\nu=i)}$	14.7654	1.1288	0.013825	8.8×10^{-7}

Table 9.
CV-residual vector.

	CV-Jacobian matrix dimension $2m \times 2n$	**Algorithms**	**Number of iterations**	**Time/ iteration (s)**	**Total time (s)**
IEEE-14	60×28	$\mathbf{3.CV - NRM^{(r)}}$	**3**	**0.045303**	**0.108547**
IEEE-30	124×60	$\mathbf{3.CV - NRM^{(r)}}$	**3**	**0.067118**	**0.158415**
IEEE-118	368×236	$\mathbf{3.CV - NRM^{(r)}}$	**8**	**0.150205**	**1.064942**
SIN-340	1704×680	$\mathbf{3.CV - NRM^{(r)}}$	**7**	**0.992872**	**6.724038**
SIN-1916	9642×3832	$\mathbf{3.CV - NRM^{(r)}}$	**8**	**29.856246**	**242.695989**

(r)—rectangular coordinates; tol.—1×10^{-3}.

Table 10.
Performance in larger systems—overdetermined matrices.

The residual vector and the chi-squared index throughout the iterations are presented in **Table 9**. Notice that the estimated values obtained in the $\mathbb{C}$-domain are exactly equal to those extracted from the power flow report in the $\mathbb{R}$-domain. Furthermore, the Gauss-Newton iterative algorithm converges in three iterations in both vector spaces, i.e., real and complex domain.

5.2 Performance in larger systems

In the sequel, **Table 10** presents the performance on larger systems of the Gauss-Newton method in complex plane which is highlighted in bold. In all simulations the convergence criterion of 1×10^{-3} is assumed.

The comparative analysis of the results presented in the aforementioned table allows us to infer that the Newton-Raphson method in complex plane has very good performance. Except for the SIN-1916 bus system, the time consuming required to achieve the solution is lower than the remainder approaches.

6. Conclusions and future developments

The chapter's prime goal is to present the advances aiming to solve nonlinear system of equations in complex plane, regardless if it is exactly or overdetermined. Firstly, it develops the former Newton-Raphson method addressed to solve exactly determined problem. Thereafter, the weighted-least-squares (WLS) is developed

through the Gauss-Newton method. In both cases the applications are carried out on small examples and larger systems. All results demonstrate the advantages of the algorithms developed in complex plane over the former procedure, although the computational burden is still a bottleneck for larger systems. Highlight that there are many enhancements that can be addressed to mitigate this problem. To cite a few, they are shown as follows:

1. Ordering schemes based on rows overlapped by columns ordering schemes [12, 13].

2. Suppressing of complex conjugate calculation storing during the Jacobian matrices building.

Nomenclature

$\underline{x_c}$	Vector of the state variables in the conjugate coordinate system
x, x^*	Complex and complex conjugate state variables
t, t^*	Complex and complex conjugate tap position
$\Re\{\cdot\}, \{\cdot\}$	Real and imaginary part of a complex variable
$\mathbf{J}$	Complex-valued Jacobian matrix for the exactly determined system of equations
$\mathbf{H}$	Complex-valued Jacobian matrix for the overdetermined system of equations
$\underline{M}$	Complex-valued mismatch vector
$(\cdot)_c$	Quantity in the conjugate coordinate system
$\|\cdot\|^2$	Squared Euclidean norm
$\|\cdot\|_\infty$	Infinity norm
ν	Iteration counter

Author details

Robson Pires
Institute of Electric Systems and Energy—ISEE, Federal University of Itajubá—UNIFEI, Itajubá, Minas Geriais, Brazil

*Address all correspondence to: rpires@unifei.edu.br

References

[1] Steinmetz CP. Complex quantities and their use in electrical engineering. In: Proceedings of the International Electrical Congress. Chicago, IL: AIEE Proceedings; 1893. pp. 33-74

[2] Kreutz-Delgado K. The Complex Gradient Operator and the CR-Calculus. San Diego, USA: Electrical and Computer Engineering—Jacobs School of Engineering; University of California; 2009. pp. 1-74. ArXIV e-print, arXIV:0906.4835v1 [math.OC]

[3] Sorber L, Van Barel M, de Lathauwer L. Unconstrained optimization of real functions in complex variables. Society for Industrial and Applied Mathematics—SIAM. 2012; **22**(3):879-898

[4] Pires R. Complex-valued steady-state models as applied to power flow analysis and power system state estimation [PhD dissertation]. Itajuba, Brazil: Institute of Electrical Systems and Energy—ISEE; Federal University of Itajuba—UNIFEI, 2018. Available from: https://repositorio.unifei.edu.br/xmlui/handle/123456789/55

[5] David AHJ. A generalization of the conjugate-gradient method to solve complex systems. IMA Journal of Numerical Analysis. 1986;**6**(4): 447-452

[6] Maltsev A, Pestretsov V, Maslennikov R, Khoryaev A. Triangular systolic array with reduced latency for QR-decomposition of complex matrices. In: IEEE International Symposium on Circuits and Systems—ISCAS. 2006. pp. 385-388

[7] Awasthi A, Guttal R, Al-Dhahir N, Balsara PT. Complex QR decomposition using fast plane rotations for MIMO applications. IEEE Communications Letters. 2014;**18**(10): 1743-1746

[8] Gentleman WM. Least squares computations by givens transformations without square roots. Journal of Mathematical Analysis and Applications. 1974;**12**:329-336

[9] Simões-Costa AJA, Quintana VH. An orthogonal row processing algorithm for power sequential state estimation. IEEE Transactions on Power Apparatus and Systems. 1981;**100**(8):3791-3800

[10] Wang JW, Quintana VH. A decoupled orthogonal row processing algorithm for power system state estimation. IEEE Transactions on Power Apparatus and Systems. 1984;**PAS-103** (8):2337-2344

[11] Vempati N, Slutsker IW, Tinney WF. Enhancement to givens rotations for power system state estimation. IEEE Transactions on Power Systems. 1991;**6**(2):842-849

[12] Catalyurek UV, Aykanat CT, Kayaaslan E. Hypergraph partitioning-based fill-reducing ordering for symmetric matrices. SIAM Journal on Scientific Computing. 2011;**4**:1996-2023

[13] Kaya O, Kayaaslan E, Uçar B, and Duff I. Fill-in reduction in sparse matrix factorizations using hypergraphs [Research Report—8448]. 2014

[14] Pires R, Mili L, Chagas G. Robust complex-valued Levenberg-Marquardt algorithm as applied to power flow analysis. International Journal of Electrical Power & Energy Systems. 2019;**113**:383-392. DOI: 10.1016/j.ijepes.2019.05.032

[15] Pires RC, Mili L, Lemos FAB. Constrained robust estimation of power system state variables and transformer tap positions under erroneous zero-injections. IEEE Transactions on Power Systems. 2014;**29**(3):1144-1152. ISSN: 0885-8950