UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Ingeniería

https://revistas.udistrital.edu.co/index.php/reving/issue/view/1185

DOI: https://doi.org/10.14483/23448393.21419

Editorial

Three-Phase Power Flow Tool for Electric Distribution Grids: A Julia Implementation for Electrical Engineering Students

Oscar Danilo Montoya[1], Alejandro Garces[2], and Walter Gil-González[2]

[1]Universidad Distrital Francisco José de Caldas. Bogotá, Colombia.
[2]Universidad Tecnológica de Pereira. Pereira, Colombia.

Most of the concepts involved in Electrical Engineering have been developed from the 20th century to the present day. One of the most popular studies in this field is the power flow problem for electrical systems composed of multiple nonlinear loads (*i.e.*, loads with constant power consumption) (1). This problem is formulated by applying Kirchhoff's voltage law to each node in the system, which results in a set of nonlinear algebraic equations that consider steady-state conditions (2). In this vein, this editorial note presents a tutorial for Electrical Engineering students regarding the general implementation of the three-phase power flow problem in distribution networks via the successive approximations method (3).

General three-phase power flow formulation

The general formula for the power flow solution in three-phase networks is obtained after applying Kirchhoff's voltage law to each network node, which produces the set of constraints defined by

$$-\mathbb{I}_{d3\varphi} = \mathbb{Y}_{ds3\varphi}\mathbb{V}_{s3\varphi} + \mathbb{Y}_{dd3\varphi}\mathbb{V}_{d3\varphi}, \tag{1}$$

where $\mathbb{Y}_{ds3\varphi} \in \mathcal{C}^{3(n-1)\times 3}$ and $\mathbb{Y}_{dd3\varphi} \in \mathcal{C}^{3(n-1)\times 3(n-1)}$ are complex sub-matrices obtained for the nodal three-phase admittance matrix $\mathbb{Y}_{\text{bus}3\varphi}$ (note that this matrix can be obtained using the node-to-branch incidence and the admittance primitive matrices (3)); $\mathbb{I}_{d3\varphi} \in \mathcal{C}^{3(n-1)\times 1}$ is the complex vector of demanded currents, which is a nonlinear function of the demanded voltages $\mathbb{V}_{d3\varphi} \in \mathcal{C}^{3(n-1)\times 1}$ and the constant complex power loads; and $\mathbb{V}_{s3\varphi} \in \mathcal{C}^{3\times 1}$ is the complex power voltage output at the terminals of the substation. Note that $\mathcal{C}$ denotes the set of complex numbers.

Editorial

*Correspondence: odmontoyag@udistrital.edu.co

Remark 1. *The calculation of the demanded current vector $\mathbb{I}_{d3\varphi}$ depends on the load connection type, i.e., Wye (Y) or Delta (Δ).*

To illustrate the general calculation of the three-phase demanded currents, consider the schematic load connections presented in Figs. 1a and 1b.

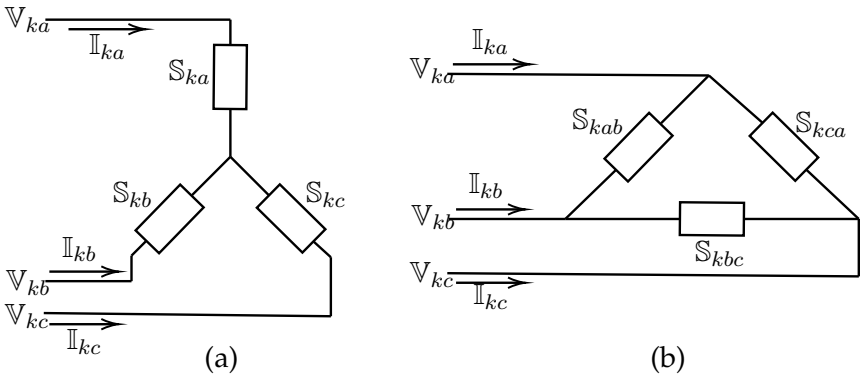

Figure 1. Constant power load: a) Y connection and b) Δ connection

For these load connections, when Tellegen's second theorem is applied (*i.e.*, the relation between voltages, currents, and powers), the following compacted three-phase current formulas are obtained:

$$\mathbb{I}^{Y}_{k3\varphi} = \mathbf{diag}^{-1}\left(\mathbb{V}^{\star}_{k3\varphi}\right)\mathbb{S}^{\star}_{k3\varphi}, \tag{2}$$

$$\mathbb{I}^{\Delta}_{k3\varphi} = \mathbf{diag}^{-1}\left(\mathbf{M}\mathbb{V}^{\star}_{k3\varphi}\right)\mathbb{S}^{\star}_{k3\varphi} + \mathbf{diag}^{-1}\left(\mathbf{M}^{\top}\mathbb{V}^{\star}_{k3\varphi}\right)\mathbf{H}\mathbb{S}^{\star}_{k3\varphi}, \tag{3}$$

$$\mathbb{I}^{\star}_{d3\varphi} = \mathbb{I}^{Y}_{k3\varphi} + \mathbb{I}^{\Delta}_{k3\varphi}, \tag{4}$$

where the matrices **M** and **H** take the following form

$$\mathbf{M} = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ -1 & 0 & 1 \end{bmatrix}, \ \mathbf{H} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix},$$

and $(\cdot)$ is the complex conjugate operation applied to the argument.

Iterative power flow solution

Considering the general power flow formula in (1) and the definition of the demanded current in (4), an iterative power flow formula can be obtained (4).

$$\mathbb{V}^{t+1}_{d3\varphi} = -\mathbb{Y}^{-1}_{dd3\varphi}\left(\mathbb{Y}_{ds3\varphi}\mathbb{V}_{s3\varphi} - \mathbb{I}^{t}_{d3\varphi}\right), \tag{5}$$

where

$$\mathbb{I}^{t}_{k3\varphi} = \mathbf{diag}^{-1}\left(\mathbb{V}^{t,\star}_{k3\varphi}\right)\mathbb{S}^{\star}_{k3\varphi} + \mathbf{diag}^{-1}\left(\mathbf{M}\mathbb{V}^{t,\star}_{k3\varphi}\right)\mathbb{S}^{\star}_{k3\varphi} - \mathbf{diag}^{-1}\left(\mathbf{M}^{\top}\mathbb{V}^{t,\star}_{k3\varphi}\right)\mathbf{H}\mathbb{S}^{\star}_{k3\varphi} \tag{6}$$

for all k in the set of the demanded buses.

Remark 2. *Eqs. (5) and (6) are iteratively solved until the desired error of convergence is reached, i.e.,*

$$\text{máx}\left|\left|\mathbb{V}_{d3\varphi^{t+1}}\right| - \left|\mathbb{V}^{t}_{d3\varphi}\right|\right| \leq \varepsilon, \tag{7}$$

with ε being the convergence's tolerance.

Julia implementation

Julia is an efficient programming environment that combines the advantages of Python, Matlab, and R with the efficiency of well-known programming languages such as C++ or Fortran (5). To illustrate the general implementation of the three-phase power flow problem in Julia, consider a small distribution grid composed of seven buses and six lines, which is unbalanced and operates with a line-to-ground voltage of 23 kV at the terminals of the substation. All the information on this system is directly presented in the Julia scripts. Fig. 2 illustrates the first part of the three-phase power flow problem's implementation in distribution networks.

```
# Successive approximation power flow three-phase grids
using DataFrames, LinearAlgebra
# System bases
Vb = 23000; # V
Sb = 1000; # kVA
Zb = (Vb^2)/(Sb*1000);
# System information Data in Ohm
branch_data = DataFrames.DataFrame([
(1, 2, 0.5025, 0.3025),(2, 3, 0.4020, 0.2510),
(3, 4, 0.3660, 0.1864),(2, 5, 0.3840, 0.1965),
(5, 6, 0.8190, 0.7050),(2, 7, 0.2872, 0.4088),]);
DataFrames.rename!(branch_data, [:k, :m, :Rkm, :Xkm])
# System information Data in kVA
node_data = DataFrames.DataFrame([
(1, 1, 0,    0,    0,    0,    0,    0,   0),
(2, 1, 1000, 600,  750,  450,  1250, 750, 1),
(3, 1, 1800, 1000, 0,    0,    900,  500, 0),
(4, 1, 3000, 1800, 0,    0,    4500, 1800,0),
(5, 1, 0,    0,    2400, 1900, 1200, 950, 1),
(6, 1, 0,    1560, 2100, 0,    1050, 780, 1),
(7, 1, 500,  0,    3000, 1150, 2500, 2300,0)
]);
DataFrames.rename!(node_data, [:k,:Vk0,:Pka,:Qka,:Pkb,:Qkb,:Pkc,:Qkc,:Type])
```

Figure 2. Parametric information associated with the three-phase 7-bus network

Fig. 2 presents the parametric information of the network, *i.e.*, the distribution line impedance (in this example, the three-phase impedance matrix is assumed to be balanced with values only in its diagonal) and the constant power loads. Note that, if `node_data.Type[k,1] == 1`, the load associated with the k-node is Y-connected; otherwise, it is $\Delta-$connected.

The next step is calculating the nodal admittance matrix. This process is shown in Fig. 3.

Once the nodal admittance matrix has been constructed, the iterative power flow formula defined in (5), the current calculation in (6), and the stopping criterion in (7) are implemented in Julia using a function called `Successive3f(node_data,Ybus3f,error,tmax,Sb)` (Fig. 4). This implementation is intuitive, and the student/researcher can easily follow all the procedures for solving

```
# Nodal admittance matrix construction
N = size(node_data,1); L = size(branch_data, 1); A = zeros(N,L)
for l = 1:L
    k = branch_data.k[l]; m = branch_data.m[l];  A[k,l] = 1;  A[m,l] = -1;
end
A3f = zeros(3*N,3*L); Z3f = complex(zeros(3*L,3*L))
z = (branch_data.Rkm + im*branch_data.Xkm)/Zb
Ybus = A*inv(diagm(z))*transpose(A)
MatrixI = diagm([1.0;1.0;1.0]);
for k = 1:L
    for j = 1:N
        if A[j,k] != 0
            A3f[3*j-2:3*j,3*k-2:3*k] = A[j,k]*MatrixI;
        end
    end
    Z3f[3*k-2:3*k,3*k-2:3*k] = z[k,1]*MatrixI;
end
Ybus3f = A3f*inv(Z3f)*transpose(A3f)
```

Figure 3. General construction of the three-phase nodal admittance matrix

the three-phase power flow problem via the successive approximations method.

Finally, to obtain the general power flow solution for the 7-bus grid used in this example, the power flow function calculates all the phase voltages, which allows obtaining the total grid apparent power losses and reporting the voltage magnitudes and angles per phase. This process is depicted in Fig. 5. It is worth mentioning that, this figure, the apparent power losses are calculated using Eq. (8).

$$\mathbb{S}_{\text{loss3}\varphi} = \mathbb{V}^{\top} \left(\mathbb{Y}_{\text{bus3}\varphi}\mathbb{V}\right)^{\star}, \tag{8}$$

where $\mathbb{V}$ corresponds to the voltage output obtained by calling the power flow function implemented in Fig. 4 as `V3f = Successive3f(node_data,Ybus3f,error,tmax,Sb)`.

The power flow solution for this numerical example is plotted in Fig. 6.

The student/researcher interested in the Julia-based three-phase power flow solution presented in this editorial note should thoroughly examine all the codifications and follow each one of the presented plots (see Figs. 2 to 6) to reach the same numerical solution presented in Fig. 6.

Conclusion

This editorial note presented an intuitive algorithm based on the successive approximations power flow method with the aim of solving the three-phase power flow problem in electric distribution networks via the Julia programming environment (version 1.9.2). The main idea of this tutorial is to provide a new power flow tool for Electrical Engineering students and researchers, an easily implementable open-source algorithm for conducting studies in unbalanced distribution networks.

```
error = 1e-10; tmax = 1000;
function Successive3f(node_data,Ybus3f,error,tmax,Sb)
    Ydd3f = Ybus3f[4:end,4:end]; Zdd3f = inv(Ydd3f);
    Yds3f = Ybus3f[4:end,1:3];
    Vs3f = node_data.Vk0[1,1]*[exp(1.0*im*0);
    exp(1.0*im*-2*pi/3);
    exp(1.0*im*2*pi/3)];
    global Vd3f = complex(zeros(3*N-3,1))
    Sd3f = complex(zeros(3*N-3,1))
    for j = 1:N-1
        Vd3f[3*j-2:3*j,1]=node_data.Vk0[j,1]*Vs3f;
        Sd3f[3*j-2:3*j,1]=[(node_data.Pka[j+1,1]+im*node_data.Qka[j+1,1])/Sb;
        (node_data.Pkb[j+1,1]+im*node_data.Qkb[j+1,1])/Sb;
        (node_data.Pkc[j+1,1]+im*node_data.Qkc[j+1,1])/Sb];
    end
    global V3f = [Vs3f;Vd3f];
    M = [1 -1 0;0 1 -1;-1 0 1]; H = [0 0 1;1 0 0;0 1 0];
    for t = 1:tmax
        Id3fY = complex(zeros(3*N-3,1)); Id3fD = complex(zeros(3*N-3,1));
        Id3f = complex(zeros(3*N-3,1));
        for k = 1:N-1
            if node_data.Type[k,1] == 1 # Wye load connection
                Id3fY[3*k-2:3*k] = inv(diagm(conj(Vd3f[3*k-2:3*k,1])))*
                conj(Sd3f[3*k-2:3*k,1])
            elseif  node_data.Type[k,1] == 0 # Delta load connection
                Id3fD[3*k-2:3*k] = inv(diagm(M*conj(Vd3f[3*k-2:3*k,1])))*
                conj(Sd3f[3*k-2:3*k,1])+
                inv(diagm(transpose(M)*conj(Vd3f[3*k-2:3*k,1])))*
                H*conj(Sd3f[3*k-2:3*k,1])
            end
        end
        Id3f = Id3fY + Id3fD
        Vdt3f = -Zdd3f*(Id3f + Yds3f*Vs3f)
        if maximum(abs.(abs.(Vd3f) - abs.(Vdt3f))) < error
            global V3f = [Vs3f;Vdt3f]
            println("Iterations: ",t)
            break
        else
            global Vd3f = Vdt3f;
        end
    end
return V3f
end
```

Figure 4. Iterative three-phase power flow solution

```
V3f = Successive3f(node_data,Ybus3f,error,tmax,Sb)
Sloss3f = transpose(V3f)*conj!(Ybus3f*V3f)*Sb
Pos = zeros(N,3);
for k = 1:N
    Pos[k,:] = [3*k-2 3*k-1 3*k];
end
Report = DataFrames.DataFrame(;k = 1:N,
MagVa = abs.(V3f[Int.(Pos[:,1])]),AngVa = angle.(V3f[Int.(Pos[:,1])])*180/pi,
MagVb = abs.(V3f[Int.(Pos[:,2])]),AngVb = angle.(V3f[Int.(Pos[:,2])])*180/pi,
MagVc = abs.(V3f[Int.(Pos[:,3])]),AngVc = angle.(V3f[Int.(Pos[:,3])])*180/pi,)
println("Complex power losses = ",Sloss3f)
println("Voltage report ",Report)
```

Figure 5. Calling the three-phase power flow function and reporting the results

```
Iterations: 6
Complex power losses = ComplexF64[425.3462475757646 + 266.4415246363203im;;]
Voltage report 7x7 DataFrame
 Row  k      MagVa     AngVa      MagVb     AngVb     MagVc     AngVc
      Int64  Float64   Float64    Float64   Float64   Float64   Float64
   1      1  1.0       0.0        1.0       -120.0    1.0       120.0
   2      2  0.989097  0.104014   0.989451  -120.079  0.987362  119.972
   3      3  0.983244  0.094141   0.987838  -120.028  0.984471  119.873
   4      4  0.979802  0.107915   0.986513  -119.977  0.98266   119.798
   5      5  0.987956  0.198423   0.986511  -120.152  0.9845    120.024
   6      6  0.985844  0.340503   0.983201  -120.318  0.981786  120.013
   7      7  0.988823  0.0813783  0.986898  -120.179  0.984177  119.931
```

Figure 6. Three-phase power flow solution in the studied 7-bus grid

Acknowledgements

The first author wants to thank the Master's Program in Electrical Engineering of Universidad Tecnológica de Pereira and the Unbalanced Distribution Grids Analysis course, wherein this material has been developed to teach graduate and undergraduate Electrical Engineering students during the second semester of 2023.

Referencias

[1] C. Yang *et al.*, "Optimal power flow in distribution network: A review on problem formulation and optimization methods," *Energies*, vol. 16, no. 16, p. 5974, Aug. 2023. ↑ 1

[2] A. Garces, "A linear three-phase load flow for power distribution systems," *IEEE Trans. Power Syst.*, vol. 31, no. 1, pp. 827–828, Jan. 2016. ↑ 1

[3] B. Cortés-Caicedo, L. S. Avellaneda-Gómez, O. D. Montoya, L. Alvarado-Barrios, and H. R. Chamorro, "Application of the vortex search algorithm to the phase-balancing problem in distribution systems," *Energies*, vol. 14, no. 5, p. 1282, Feb. 2021. ↑ 1

[4] O. D. Montoya, J. S. Giraldo, L. F. Grisales-Noreña, H. R. Chamorro, and L. Alvarado-Barrios, "Accurate and efficient derivative-free three-phase power flow method for unbalanced distribution networks," *Computation*, vol. 9, no. 6, p. 61, May 2021. ↑ 2

[5] J. M. Perkel, "Julia: Come for the syntax, stay for the speed," *Nature*, vol. 572, no. 7767, pp. 141–142, Jul. 2019. ↑ 3

Oscar Danilo Montoya

Compatibility and Electromagnetic Interference group, Department of Engineering, Universidad Distrital Francisco José de Caldas; Electrical Engineer, Master in Electrical Engineering, and PhD in Engineering. **Email:** odmontoyag@udistrital.edu.co

Alejandro Garces

Electromagnetic Fields and Energy Phenomena group, Department of Engineering, Universidad Tecnológica de Pereira; Electrical Engineer, Master in Electrical Engineering, and PhD in Engineering.
Email: alejandro.garces@utp.edu.co

Walter Gil-González

Electromagnetic Fields and Energy Phenomena group, Department of Engineering, Universidad Tecnológica de Pereira; Electrical Engineer, Master in Electrical Engineering, and PhD in Engineering.
Email: wjgil@utp.edu.co

Oscar Danilo Montoya, Alejandro Garces, Walter Gil-Gonzalez

Three-Phase Power Flow Tool for Electric Distribution Grids: A Julia Implementation for Electrical Engineering Students

Ingeniería
vol. 28, no. 3, e21419, 2023
Universidad Distrital Francisco José de Caldas,
ISSN: 0121-750X
ISSN-E: 2344-8393

DOI: https://doi.org/10.14483/23448393.21419