9th International Conference. Electrical Power Quality and Utilisation. Barcelona, 9-11 October 2007

The Load Flow Calculation in Unbalanced Radial Electric Networks with Distributed Generation

Mircea Chindriş
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
e-mail: mircea.chindris@eps.utcluj.ro

Antoni Sudrià i Anderu
Universitat Politècnica de Catalunya
Barcelona, Spain
e-mail: sudria@citcea.upc.edu

Ciprian Bud, Bogdan Tomoiagă
National Power Grid Company - Transelectrica
Transport Branch Cluj
Cluj-Napoca, Romania
e-mail: ciprian.bud@transelectrica.ro
bogdan.tomoiaga@yahoo.fr

***Abstract*—Besides industrial or domestic customers, some single-phase distributed generators can impose an unbalanced operation of electrical networks. In the case of balanced power systems operated in radial schemes, the load flow calculation can be performed using a particular method known as the backward/forward sweep. For unbalanced radial electrical networks, we can consider the backward/forward sweep algorithm with some particular adaptations. For this purpose, authors propose to model the electrical three phase quantities as abstract data types characterized by three parameters and some definite operations. By implementing this model, the computation of power flow in unbalanced power systems is reduced to the evaluation of some simplified expressions. To illustrate the proposed methodology, authors have developed original software that allows performing all necessary operations with three-phase unbalanced electrical quantities.**

Keywords-unbalanced electric networks, abstract data types, object oriented programming, power flow, distributed generation.

I. INTRODUCTION

Besides industrial or domestic customers, some single-phase distributed generators can impose an unbalanced operation of electrical networks.

The load flow calculation for unbalanced electrical networks consists in the determination of steady state quantities for each phase at a time. The theoretical relationships proposed in literature are difficult to be implemented in complex systems containing many elements, in various combinations.

Nowadays, besides data types existing in the number theory, the high level programming languages allow the definition of new artificial data types, for instance *abstract data types*.

The term *type of data* designates a *set of values* (the domain of type) and *a set of operations* that can be performed with these values. The set of operations can be divided in three subsets:

- operations among the same type of data;
- operations among the specified type of data and an other type of data;
- operations performed on the data itself.

As an example, we can consider data belonging to the real numbers set (the domain of type). In this case, they can perform arithmetical operations with another real number (operations among the same data types), arithmetical operations with an integer number (operations among the specified type of data and another type), and the extraction of integer part (operation applied to the data type itself) [1].

In this paper, the authors propose to model the three-phase complex quantities (symmetrical, asymmetrical, balanced or unbalanced) through *abstract data types* with three complex parameters (*r, s, t*); among these parameters, some operations are defined.

By implementing this model, all three-phase quantities will be considered as “complex three-phase” objects. As a result,

electrical engineering laws, as Ohm's or Kirchhoff's, are reduced to simplified expressions corresponding to single-phase case.

To specify an actual *abstract data type*, it is necessary to specify the two elements of the type, i.e. the domain and the operations:

- *the domain* is specified as a mathematical set;
- *an operation* is described by its mathematical definition.

To illustrate the proposed methodology, authors have developed original software that allows performing all necessary operations with three-phase unbalanced electrical quantities.

II. Electric Quantities as Abstract Data Types

A. Complex Quantities as Abstract Data Types

Complex numbers are very frequently used in electrical engineering. They have the following characteristics:

1) Domain: The domain is indicated by *C* and is specified as:

$$C = \{(a,b) \mid a,b \in R\} \tag{1}$$

2) Operations Set: In the complex numbers set, the following operations are defined:

- for any two quantities $v_1, v_2 \in C$, it is possible to calculate the quantity $v \in C$, with parameters a_e and b_e, defined through:

  Addition: $v = v_1 + v_2$, where:

  $$a_e = a_1 + a_2, \quad b_e = b_1 + b_2 \tag{2}$$

  In a same manner, it is possible to define the operations: subtraction (-), multiplication (*) and division (/).
- for any quantity $v \in C$, it is possible to compute the subsequent quantities:

  Conjugate: $v_1 \in C$ defined through $v_1, v_2 \in C$, with parameters a_e and b_e calculated as follows:

  $$a_e = a_1 \quad b_e = -b_1 \tag{3}$$

  Module: $r \in R$ defined by $v = v_1 + v_2$, calculated as:

  $$r = |v| = \sqrt{a^2 + b^2} \tag{4}$$

  Angle: $\theta \in R$ defined by $\theta = Angle(v)$, calculated as: $\theta = Angle(v) = atg\frac{b}{a}$ (5)

B. Three Phase Quantities as Abstract Data Types

1) Domain: The domain of the complex three-phase numbers is denoted with *CT* and is specified as:

$$CT = \{(r,s,t) \mid r,s,t \in C\} \tag{6}$$

2) Operations Set: The following operations are defined in this set:

- for any two quantities $v_1, v_2 \in CT$, it is possible to calculate the quantity $v \in CT$, with parameters r_e, s_e and t_e, defined through:

  Addition: $v = v_1 + v_2$, where:

  $$r_e = r_1 + r_2 \quad s_e = s_1 + s_2 \quad t_e = t_1 + t_2 \tag{7}$$

  In a same manner, it is possible to define the operations: subtraction (-), multiplication (*) and division (/).
- for any quantity $v_1 \in CT$, it is possible to determine the quantity $v \in CT$ with parameters r_e, s_e and t_e defined through:

  Symmetrical components: $v = SC(v_1)$, where:

  $$\begin{aligned} r_e &= \frac{1}{3} \cdot (r + s + t) \\ s_e &= \frac{1}{3} \cdot (r + s \cdot a + t \cdot a^2) \\ t_e &= \frac{1}{3} \cdot (r + s \cdot a^2 + t \cdot a) \end{aligned} \tag{8}$$

  where r_e represents the zero sequence component, s_e represents the positive sequence component, and t_e represents the negative sequence component.

  The following parameters can be also easily computed:

  - *Unbalance factor*:

    $$k^- = |t_e| / |s_e| \tag{9}$$

  - *Asymmetry factor*:

    $$k^0 = |r_e| / |s_e| \tag{10}$$

- for any quantity $v_1 \in CT$ it is possible to determine the quantity $c \in C$ defined through:

  Equivalent vector: $hc = \overline{v}_1$, calculated as:

  $$hc = r_1 + s_1 + t_1 \tag{11}$$

III. Load Flow Calculation For Unbalanced Radial Electric Networks

The load flow calculation for electric networks consists in the determination of steady state quantities associated to its elements. The radial configuration is particularly used in electric distribution networks.

A. Adaptation of backward/forward sweep algorithm for unbalanced radial electric networks

In this case, we have two types of nodes:

- one source node (infinite bus), to which the specified quantities are the components of voltage, and the unknown quantities are the components of loads (powers, currents);
- more consumer nodes, to which the specified quantities are different loads (modelled as constant powers, constant currents, constant impedances/admittances or a combination of the above elements), and the unknown quantities are the components of the voltage. The capacitor banks can be represented in a same manner: as constant impedances in a combination with the specified characteristics of loads.

The electric lines are considered as balanced or unbalanced impedances.

The load flow calculation can be performed using a particular method referred to as the *backward/forward sweep*. This method consists of two steps [2]:

- *Backward sweep,* where, starting from the end nodes and going towards the source node, and using the Kirchhoff's current law, the current at each load node, as well as the current flowing through its ingoing branch, are calculated*;*
- *Forward sweep,* where, starting in the opposite direction, from the source node *S* (whose constant voltage is taken as reference) and going towards the end nodes, using the Ohm's law, the voltage drop on each branch, as well as the voltage at each load node, are calculated.

In unbalanced power systems, the above-proposed models can be introduced. Accordingly, the load flow calculation algorithm using the backward/forward sweep consists in the following steps:

a. Ordering the network (indexing the ingoing node and branch for each load node) and setting the voltages at the load nodes to the value of the source node (S) phase voltage – in this first step, the voltage system is considered as balanced:

$$U_k^{(0)} = U_S, k = 1,2,\ldots,n, k \neq S \tag{12}$$

where: $U_S, U_k \in TC$;

b. Setting the initial iteration index: $p = 1$;

c. *Backward sweep:* crossing the network from the end nodes towards the source node and performing the following operations:

c.1. Calculation of the current at the node *k* by using the expression of the load power given by:

$$I_k^{(p)} = \frac{S_k^*}{U_k^{*(p-1)}} \tag{14}$$

where $I_k, S_k^* \in TC$ (in this case loads are represented only through constant powers);

c.2. Calculation of the current flowing through the branch ingoing to node *k:*

$$I_{ik}^{(p)} = I_k^{(p)} + \sum_{j \in Next(k)} I_{kj}^{(p)} \tag{15}$$

if *connection* = Y → $I_{ikN}^{(p)} = \bar{I}_{ik}^{(p)}$ (16)

if *connection* = Δ → $I_{ikN}^{(p)} = \sum_{j \in Next(k)} I_{kjN}^{(p)}$ (17)

where: *i* is the index of the node up stream to the node *k;*
Next(k) - the set of nodes next to the node *k;*

d. *Forward sweep:* the calculation of voltages at the nodes, crossing the network from the source node towards the end nodes. For the in progress iteration *p,* considering the crossing direction of a branch from the node *i* towards node *k,* the calculation is performed in the following manner:

d.1. Calculation of the voltage drop on the *i* - *k* branch:

$$\Delta U_{ik}^{(p)} = Z_{ik} \cdot I_{ik}^{(p)} + Z_{ikN} \cdot I_{ikN}^{(p)} \tag{18}$$

where: $Z_{ik} \in TC$;

$Z_{ikN} \in C$ - impedance of neutral conductor;

d.2. Calculation of the voltage at the node *k:*

$$U_k^{(p)} = U_i^{(p)} - \Delta U_{ik}^{(p)} \tag{19}$$

e. Calculation of the power injected into the network by the source node:

$$S_S^{(p)} = U_S \cdot \sum_{i \in Next(S)} I_{Sj}^{*(p)} \tag{20}$$

f. If $p > 1$ and $| \bar{S}_S^{(p)} - \bar{S}_S^{(p-1)} | \leq \varepsilon_S$ then go to the next step, else update $p = p + 1$ and go to step c.

g. Ca1culation of power losses through the network branches.

B. Adaptation of backward/forward sweep algorithm for unbalanced radial electric networks with distributed generators

Distributed generators can be classified as follows:

- generators which provide *constant power* (*PQ nodes*), to which the specified quantities are the imposed active and reactive power and the unknown quantities are the components of voltage;
- generators which provide *constant active power* and *constant module of voltage* (*PU nodes*), to which the specified quantities are the active power and module of voltage, and the unknown quantities are the reactive power and the angle (phase) of voltage. These sources can generate active power and sometimes can generate or consume reactive power, having the possibility to maintain the nodal voltage at a set value by means of an automatic voltage regulator.

For unbalanced radial electric networks that include distributed generators, we can consider the backward/forward sweep with some conditions and adjustments. The algorithm is as follows [2]:

1. Initialise the iterative step $p = 0$ and establish the initial value of the (generated) reactive power for every *PU* nodes: $Q_{g,k}^{(0)} = 0$;
2. Update the iterative step $p = p + 1$;
3. Perform load flow calculation by *backward/forward sweep*;
4. For each node *k* (*PU* type), if $\| [U_k^{(p)}]_{phase} | - | [U_k^{specified}]_{phase} \| \leq \varepsilon_U$ or $Q_{g,k}^{calc} = Q_{g,k}^{\min}$ or $Q_{g,k}^{calc} = Q_{g,k}^{\max}$, the iterative process stops;
5. Calculate the generated reactive power $Q_{g,k}^{calc}$ necessary to achieve the specified voltage $U_k^{specified}$ at a node *k* (the *PU* nodes), e.g. using the secant method:

$$Q_{g,k}^{calc} = Q_{g,k}^{(p-1)} + \frac{Q_{g,k}^{(p-1)} - Q_{g,k}^{(p-2)}}{|U_k^{(p-1)}| - |U_k^{(p-2)}|}\left(|U_k^{(p)}| - |U_k^{specified}|\right) \quad (21)$$

$$\begin{cases} if \quad \left(Q_{g,k}^{\min} \le Q_{g,k}^{calc} \le Q_{g,k}^{\max}\right) \quad then \quad Q_{g,k}^{(p)} = Q_{g,k}^{calc} \\ if \quad \left(Q_{g,k}^{calc} < Q_{g,k}^{\min}\right) \quad then \quad Q_{g,k}^{(p)} = Q_{g,k}^{\min} \\ if \quad \left(Q_{g,k}^{calc} > Q_{g,k}^{\max}\right) \quad then \quad Q_{g,k}^{(p)} = Q_{g,k}^{\max} \end{cases} \quad (22)$$

6. Go to step *2*.

By introducing the objects defined in chapter II in the backward/forward sweep algorithm for balanced networks (with demonstrated correctness and convergence) [2], we obtained a novel algorithm for load-flow calculation in unbalanced operation states. It is important to specify that the number of iterations is a function of two constants indicating the precision of calculus: ε_S (the deviation of injected power in the source node – infinite bus) and ε_U (the deviation of voltage in *PU* type nodes).

IV. APPLICATION

The high level programming languages (Object Pascal, C++, Java etc.) allow the definition of *abstract data types* as objects and the implementation of the above-defined models. The authors have selected the *C++* language program because it assures the overcharge of existing arithmetical operators (they are redefined in the program).

Figure 1 presents the object (*class*) *C* (complex), the member functions implementing the relationships presented in chapter II.A. *This unit can be used in any program written in C++ language for all calculus with complex numbers* [4].

```
typedef double REAL;                    //define a real number
class C                                 { //class complex
 protected:                             //the protected part
    REAL a, b;                          //real and imaginary data
 public:                                //the public part
    C();                                //constructor
    ~C();                               //destructor
    virtual C operator + (C& c);        //operation of addition
    virtual C operator - (C& c);        //operation of subtraction
    virtual C operator * (C& c);        //operation of multiplication
    virtual C operator / (C& c);        //operation of division
    virtual C operator * (REAL r);      //operation of multiplication
                                        //with a real number
    virtual C conjugate();              //operation by conjugate
    virtual REAL module();              //module operation
                                       };
```

Figure 1. Complex object implementation

Figure 2 presents the object *TC* (complex three-phase), the member functions implementing the relationships presented in chapter II.B. *This unit can also be used in any program written in C++ language*.

Based on these two units (*C* and *TC*), authors have developed an original program, named *OBthree*, dedicated to mathematical operations with three-phase quantities. Materialized with *GUI* (*Graphic User Interface*) technology, the program ensures an extreme lightness in the use.

```
#include "TAD_C.cpp"//              //include C object
class TC                           { //class three complex
 private:                          //the private part
    C r, s, t;                     //three complex data
 public:                           //the public part
    TC();                          //constructor
    ~TC();                         //destructor
    TC operator + (TC& tc);        //operation of addition
    TC operator - (TC& tc);        //operation of subtraction
    TC operator * (TC& tc);        //operation of multiplication
    TC operator / (TC& tc);        //operation of division
    TC operator * (REAL r);        //operation of multiplication
                                   //with a real number
    TC operator + (C c);           //operation of addition
                                   //with a complex number
    TC conjugate();                //operation by conjugate
    C equivalent_v();              //equivalent vector operation
    TC symmetrical_components();   //symmetrical components operation
    REAL assym_factor();           //asymmetry factor
    REAL unb_factor ();            //unbalance factor
                                  };
```

Figure 2. Complex three-phase object implementation

Any three-phase complex quantity is introduced through *Input* panel and the *equivalent vector* (VEchiv), *zero sequence component (H), positive sequence component (D), negative sequence component (I), asymmetry factor (K0) and unbalance factor (K-)* are automatically calculated – Figure 3. It is possible to rotate (electrical rotation) a three-phase object by clicking on the button *r* or to obtain its *conjugate* (by clicking on the button *c*).

The following objects are defined in the program:

- *A, B, C, D, E, R, G*: complex three-phase quantities, which are updated from the *Input* panel with the "**>**" button;
- *I, i*: complex quantities;

In a same manner, ten locations (*X, Y, Z, V, W, x, y, z, v, w*) are defined for writing relationships among the complex three-phase quantities (objects). For every relationship, the result appears also in the *Input* panel by making a "click" on object's symbol (for example Y).

From Figure 3, it can be seen that in an unbalanced electric power system, the voltage drop can be obtained by performing arithmetical operations with proposed objects describing line currents (object *B*), line impedances (object *C*), neutral current (object *I* corresponding to the geometrical sum of the three line currents - equivalent vector), and neutral impedance (in this case purely resistive *10 ohm*). Practically, the voltage drop on each phase (object *Z*) can be obtained by adding the complex voltage drop on the corresponding phase (object *X*) to the voltage drop on the neutral conductor (object *Y*).

V. CONCLUSIONS

The paper presents an original solution to perform calculus with complex and three-phase complex quantities by using *abstract data types*. Related mathematical models for *complex abstract data type* and *three-phase complex abstract data type* are developed. These structures were implemented as objects in original software which allows manipulating complex and

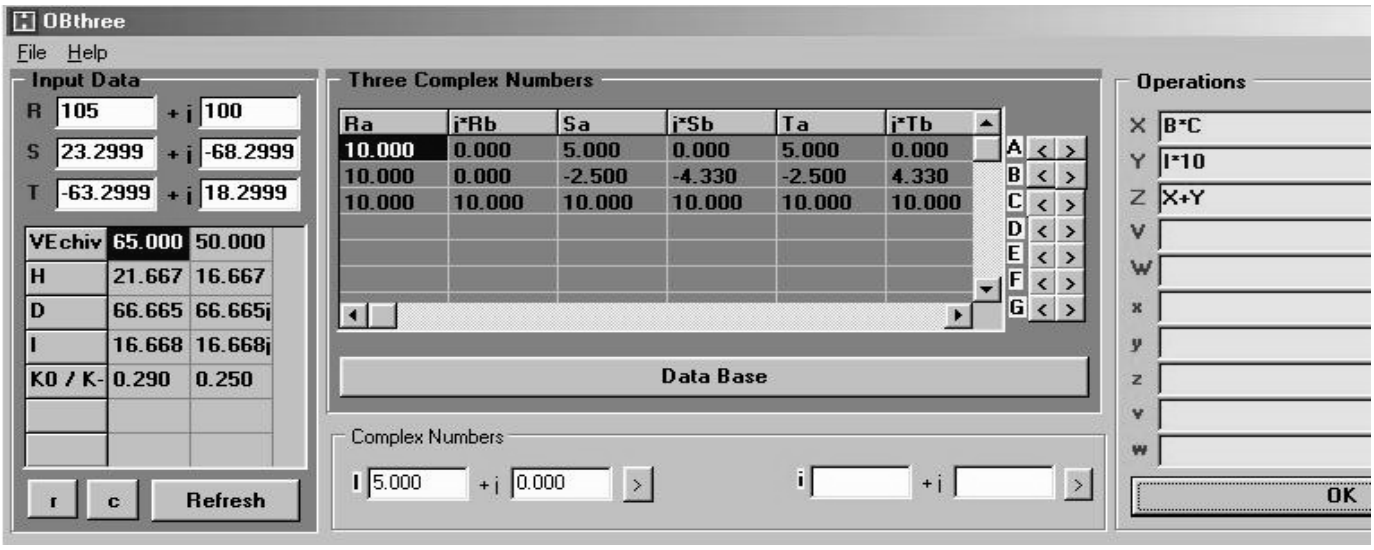

Figure 3. OBthree program

three-phase complex quantities by simply using common operators like "+", "-", "*" and "/".

By implementing these objects in a backward/forward sweep algorithm for balanced network with distributed generation, a new algorithm for unbalanced regime was proposed in the paper.

A numerical example is given to confirm the computer implementation of proposed objects; the complete algorithm for the power-flow computation is under a test process for validation and will be presented in future works.

The authors intend to develop an algorithm and an associated program for load-flow analysis in both unbalanced and harmonic polluted electric networks with distributed generation.

ACKNOWLEDGMENT

The authors wish to thank National Power Grid Company – Transelectrica for the financial support.

REFERENCES

[1] M. Chindriş, B. Tomoiagă, C. Bud, "A new object oriented program for unbalanced power systems analyse", in: 4-th International Conference on Electrical and Power Engineering - EPE 2006, Iaşi, 12-14 October 2006, pp. 1343-1348.

[2] M. Eremia et alii, Electric Power Systems – Electric Networks, vol. I. Bucuresti: Academiei, 2006.

[3] V. Cioban, Z. Darvay, The programming evolutes methods – C and C++ languages (in romanian). Cluj-Napoca: Litho. Univ. "Babes-Bolyai", 2001.

[4] B. Stroustrup, C++ Programming Language – Special Edition (in romanian). Bucuresti: Teora, 2003.