Advances in Computer Science and Engineering
Volume 9, Number 1, 2012, Pages 17-33
Published Online: November 2012
Available online at http://pphmj.com/journals/acse.htm
Published by Pushpa Publishing House, Allahabad, INDIA

A MULTI-AGENT BASED PLATFORM FOR SYSTEM ANALYSIS IN ELECTRIC POWER NETWORK IMPLEMENTED WITH FRACTIONAL NUMBERS

Jean-Pierre Lienou, Maurice Tchuente and Emmanuel Tanyi

National Advanced School of Engineers, UY I
Department of Computer Engineering, IUTFV
UDS, LIMSS, LAIA Yaoundé, Cameroun
e-mail: lienou@gmail.com

Faculty of Sciences
University of Yaoundé I
Yaoundé, Cameroun
e-mail: maurice.tchuente@gmail.com

Faculty of Engineering
University of Buéa
Buéa, Cameroun
e-mail: emmantanyi@yahoo.com

Abstract

Tasks related to the analysis of electric power network analysis are daily tasks for engineers of the power grid and fortunately, several tools are provided to carry out the specific tasks such as: load flow problem (LFP), the calculation of short-circuit analysis and stability studies. Those tools lack flexibility and sometimes precision that is needed to plan or assess the power network in real time. There is a need of an intelligent tool to help operators to take important decisions

Keywords and phrases: component, load flow problem, fractional number, multi-agent systems, Jademx.

Received June 30, 2012

to act as fast as possible to protect the power network. Power network is a highly non-linear and complex system which is considered as a critical infrastructure for any society in the will to self-development and that should be accordingly supervised. In this paper, there is an implementation of several power network analysis tasks using intelligent agents to speed up decision processes and the precision in several algorithms is increased by replacing real numbers with two integers. A tip is used to limit up the number of digits used to approximate real numbers. The obtained results show that studied algorithms converge faster and the learning mechanism used allow to select the right method of calculating the load flow problem, build dynamic topology and compare with topologies that occurred in the past. The platform is implemented using Java Enterprise technology with Eclipse Helios, the Jademx plug-in of the Jade middleware.

I. Introduction

Power grids of developing countries are scattered in large geographical areas, and they show a high level of intermixed technologies at the production level. They are to merge to form larger networks with distributed production sites at regional levels. This interconnection is only possible if the parameters of the different networks are compatible. In such circumstances, the calculation of LFP is a task that is almost daily task for two major categories of network engineers. The network planning engineers plan the production of electricity while maintenance engineers use to make decisions on operational duties in order to maintain the network in stable condition. Many tools are available for such processing and the calculations are often made in “per unit” and the representation of real numbers in modern computers is usually in IEEE 754 format. It means that the accumulated errors do not give a true and reliable solution as the computation lasts. In addition, the results are available only in remote control centers.

Short-circuit faults are very prompt and their consequences on the network may be very harmful. To take the right decision on how or which network the operators should act in just few minutes or less, it is important to develop a tool that may help them in the decision making process. The electrical boxes are often subject to this type of failure because, the heating

conductors are dilated, can melt the insulation and thereby creating short-circuits that can be very expensive to repair.

The LFP is equivalent to calculate the voltage and angle at each network node, or to calculate the active and reactive power generating sources in the network. Among the most used algorithms to solve the LFP, we can mention: Newton-Raphson both in polar and Cartesian coordinates which is widely used in commercial software, the Fast Decoupled Algorithm, the Jacobi and Gauss Seidel algorithms, and those based on nodal matrices.

Known techniques for solving systems of equations are: the inversion of matrices, the substitution of variables or an iterative method. According to the network parameters, the matrices may lead to a divergent solution or slowly convergent one. This raises the question: which is the best algorithm to solve the LFP under certain constraints set by the network engineer? These constraints can be: the number of network nodes, the time taken to calculate the desired precision, the memory used, the ability to reuse the results and so on.

This paper is twofold: to propose a multi-agent system for choosing the method of calculating analysis operations (LFP, short-circuit, stability study) and to show how to improve existing methods by minimizing errors accumulated in the “per unit” system using fractional numbers instead of real numbers.

The paper is structured as follows: in Section II, we present a comprehensive review of LFP algorithms and a background on MAS. Section III discusses on the advantages and drawbacks of using fractional numbers and the environment in which it might be benefitted to implement. Especially, there are ways to an FPGA implementation of the solution to free up valuable CPU time on machines to allow a quick calculation. In order to comply with the CIM, a discussion on how to map it to the RDBMS MySQL is provided. In Section IV, we present the multi-agent system that allows the system to always select the best solution to the LFP, to calculate short-circuit and the learning process of the agents. The solutions are obtained by unsupervised learning on the previous states of the network for which we

made an analysis and the outcome is satisfactory to an expert. The results are presented in Section V, we end up with a conclusion and future work.

II. A Review of LFP Algorithms, Software and Multi-agent Systems

Several formulations for solving LFP exist and they can be grouped into three categories: mainly, those based on Gauss-Seidel, Newton-Raphson and Backward/Forward sweep. In [15], a didactic case is used on a network of five nodes for almost eight different algorithms. Various implementations can optimize speed, memory used, fast convergence, etc. as criteria to choose LFP method in a given network.

The “Sweep Backward/Forward method” is popular for its processing speed and is convenient for low-meshed distribution networks. This type of algorithm is the basis for the iterative solver implemented in CYMDIST software. Newton-Raphson’s methods are based on injecting power in nodes tend to stand out by their performance in handling large meshed systems. Gauss-Seidel methods on the other hand, are less robust and generally more difficult to converge but are usually used to compare algorithms.

The simulation tool EMTP-RV uses a formulation based on the relations which differ from the current algorithms of Newton-Raphson (NR) used in network analysis software. This formulation allows describing systems in matrix form called “Modified Nodal Analysis” (MNA) in an augmented matrix form [1, 4]. The Multi-Area Thévénin Equivalent (MATE) is used in the Object Virtual Network Integrator (OVNI) simulator in [8].

The method is built on relationships among currents, (Kirchoff’s Laws) instead of the relationship of power, which simplifies the modeling of network components, but also the representation of connections between components (delta, triangle, phase to phase, etc.). The technique allows deriving the Jacobian matrix in a simplified form and the linear system resulting matrices are sparse. The expression of the system is in rectangular coordinates (real/imaginary) and the partial derivatives are expressed in relation to voltage and current of the network node and its components. This method allows to model networks with their actual data without having to

express the values in "per unit." It is then possible to change the ratios of transformation and also the operating voltages in the system, without having to re-evaluate the admittances in network matrix as new VA_{base} of the system, as is the case when working in the per unit system. Recall that over three quarters of methods use the "per unit" system [6, 9]. It is retained in this paper the three most powerful regardless of network topologies: that of NR in Cartesian coordinates, those of the Fast Decoupled in polar coordinates and the MNA. The Jacobi method is introduced for comparison purposes. The pseudo code and flow charts of these methods can be found in [15] along with the simulation results used to compare with our approach. Many tips and techniques are provided in [8] on how power system computation should be optimized. Due to intensive computation of those algorithms, we realized that it should be benefitted in terms of precision to change the representation of all floating points with fractional numbers, in order to minimize errors especially when using "per unit" system. The next paragraph explains how it has been done.

A. IEEE 754 format, conversion from real to fractional numbers

Nowadays, computers represent real numbers in IEEE 754 format. Remember that representation even when the number is fractional is approximate. For example, the number 3.18 cannot be represented without error as it is known that can be represented without error only rational numbers which can be represented in the form of $X/2^k$. IEEE 754 format is used for the representation of real numbers either in single or double types and respects the format: "sign + Exponent (biased $2^{k-1}-1$, where k is the number of bits in the exponent) + Mantissa". In scientific representation, the bit before the comma is usually omitted ("the hidden bit"). In Intel based microprocessors, we have the extended type which is used to compensate shortfall during arithmetic operations and there is no "hidden bit".

B. Processing steps used in this work

A class in Java was developed to convert real numbers into a set of two integers: a numerator and a denominator. If necessary, a Euclidean algorithm

is applied to find the Greatest Common Denominator in order to find the lowest terms. From that point, the number $\frac{1}{10}$ which is represented with a period [0011] will be more accurate than the real version. Other methods were used to compare, invert, add, multiply or divide fractions as we could not find out in Java, the classical overloading operators encountered in C++. The conversion method `asDouble ()`, allows us to return to the real type for display purposes.

Having the class fraction written, the class Nombre-Complexe is developed and it enables to use fractions to represent the real and imaginary parts of complex numbers. Specific methods such as: conjugate(), multiply() replacing the operators are written to help manipulate complex numbers.

C. Agent-oriented technology and programming overview

Agent technology is considered by many specialists, as the solution to the legacy software integration and as a paradigm that may improve current methods for conceptualizing, designing and implementing software systems [3]. It has been used to develop even hardware, especially in the telecommunication area. Without diving deeply into what is an agent, we consider it as a special component (not only software) that has autonomy, working for some “clients” in pursuit of its own agenda; it is social, reactive, truthful, proactive and benevolent. There are many other properties that can be found in the literature [3, 7, 14]. A multi-agent system (MAS) is a set of agents working together. MAS can be defined as: a network of interacting software modules that bring together dispersed systems that collectively manage complex tasks that are beyond the capacity of any individual system on the network.

The four main architectures of MAS are: reactive, layered, logic-based, and BDI (belief, desire, and intention).

The way they work together raises a lot of technical problems related to communication and how to coordinate their work. The main two usually used languages are KQML and FIPA-ACL. The coordination tasks are carried out

through organizational structures of agents. The coordination technique may be achieved through the use of protocols which may be decentralized, centralized or distributed where negotiation is usually used.

There are many agent oriented platforms that provide means to deploy MAS on various hardware and environment. Most of the time, the languages developers used within those frameworks are C# and Java. The platform used here is Jade which uses Java.

D. Power systems and MAS

MAS have been used in several areas of power network from protection, control, diagnostic, supervision to supply demand matching and power restoration.

In [2], authors used MAS to solve problems of power restoration and fault isolation, implementing it on power line communication. Authors of [16] proposed to solve the problem of centralized operations in the medium voltage part of electric power network, since deregulation of the electricity market makes the pertinence of such an organization not ensured. In [12], MAS is used to apply a cost-scaling push-**relabel** algorithm in order to solve the OPF in a distributed agent environment. It helps power flow management that would need to be distributed, flexible and intelligently, in order to cope with the challenges on bi-directional power flow in the network, and the uncertainty in the forecast of power generation from grid connected renewable and distributed energy sources. In [5], the ICT requirements are examined in order to implement MAS in power supply and demand matching. In [11], the IEEE provide guides on how to develop MAS especially in power electric network.

III. On the Use of Fraction Number in Power Analysis Algorithm and CIM Implementation Standard

In our software, the impedances of the lines are usually given in “per unit”. As a case study, we took two networks that LFP simulations are well studied and available with several algorithms. We choose the five-node

network described in [15] and the Southern Cameroon interconnected network of 25 nodes, studied in many Master Theses at the Advanced National School of Engineers.

The admittance matrix of the systems was built using the fractional number system and compared to computation of the same matrix using numbers in double format and we found that, in many cases, errors have been found at the first decimal place. An error is detected in the calculation of this matrix in the case of network nodes in 5 [15]. The error found is that, the admittance matrix was not symmetric due to the fact element (2, 3) of the matrix, is the conjugate of (3, 2) for the lines connecting nodes 2 and 3. The comparison with the simulation of the RISC, took data from a Matlab code simulation and the errors were in the same margins as those from the five nodes system.

A. Advantages, disadvantages of the approach and proposed solutions

The use of fractional numbers is intended to accelerate calculations, and increase accuracy of results without using a specialized processor for floating point calculations. It is assumed that, the calculations on integers are faster than on type `double`. Remember that, the same precision on a real number coded on n bits can be obtained by coding the numerator and denominator with $n/2$ bits. Systems are traditionally given in "per unit" and a fractional conversion requires taking several decimal places to maintain accuracy. As a benefit, the system only works on integers of type long and it is easily implemented in a FPGA that can be placed in a strategic asset on the power grid [9, 10]. The accuracy is higher due to the fact that, the intermediate calculations are done on actual rounded values and it is the main source of errors in the calculation method with the double type.

We identified two major drawbacks. The first drawback of our approach is that, numerator or denominator can grow very quickly and can exceed the limits of the type long as in our case in Java for a small real. The second drawback is that, Euclid's algorithm is often called to make the fraction irreducible. We solve the first problem by reducing intelligently the

precision. We use two tricks to avoid the problem: first, in the Java code, we found that, when the numerator and denominator figures were over nineteen, it was likely that future operations are not correct. The solution lies in reducing the number of bits allocated to the numerators and denominators. But, since this reduction leads to a loss of accuracy, we performed the operation we propose to use an agent embedded in an FPGA for fractional calculations. In [9], the authors show that, we can solve this problem by using *n* up to 35.

B. Use of CIM and encountered problems

For the analysis of the network, we tried to meet up the Common Information Model (CIM). A list of implementation problems was encountered.

The first attempt was to use CIMTOOL to build an application respecting the information system proposed by the IEC. Version 15 was used while Java and SQL models have been obtained from the file iec61970cim14v15.xmi available on the Internet. There were several problems when mapping the SQL model in the RDBMS MySQL. The execution of the script in MySQL Workbench has set the following problems:

- The names of the relations are generated in double quotes and that is an unusual situation. To solve the problem, a global replacement was made.
- Fields types are generated with the type string CHAR VARYING unrecognized in the current version. We replace all these types with VARCHAR.
- One of the tables to be created is the table “LIMIT”, which is a keyword of the SQL language in MySQL 5.5. The temporary solution is to create the table “limite” rather. This reduces the degree of compatibility of the base.
- The tables unitmultiplier and (unitmultiplier 1 and unitsymbol

unitsymbol1) have the declaration of the field name as a string. However, the single field does not sort in the light of the case. The unit Henry noted ‘H’, is incompatible with the time denoted by ‘h’. It is the same for multiple Mega rated ‘M’ and milli noted ‘m’. To solve the problem, VARCHAR has been transformed into VARBINARY type for these tables. We decide not to solve the problem by using the collation language mechanism implemented in MySQL.

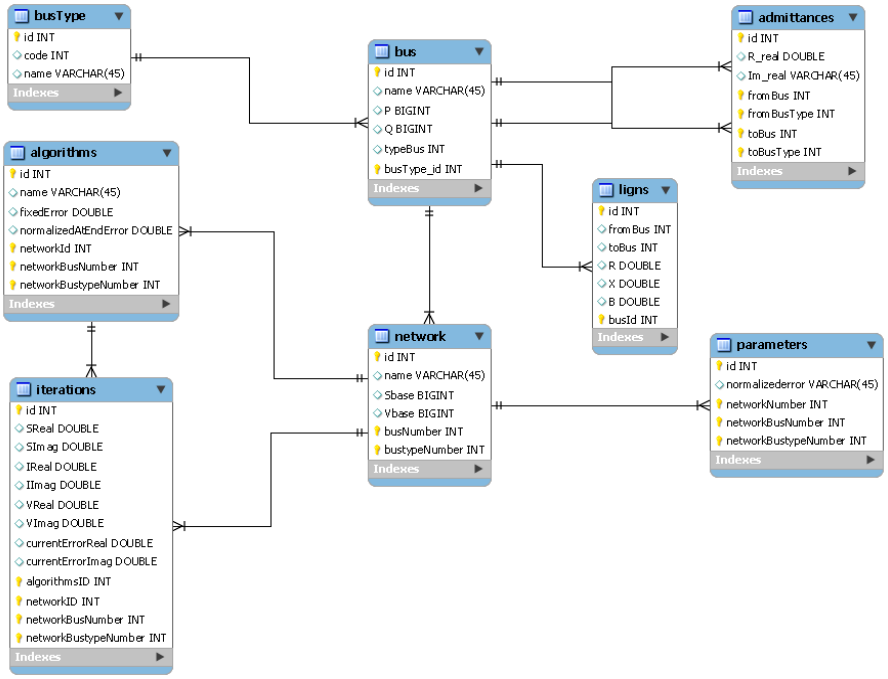

Figure 1. Structure of the database used as a subset of CIM and rename for comfortable purpose.

For a simple network application, we have not considered interesting to take a basic CIM compatible because of its weight and highly distributed information in the database. We have preferred a small model valid for specific applications to the study of LFP and that can be mapped into the CIM compatible software.

The class diagram in Figure 1 shows the entity relationship model adopted. The relation “ligns” show that, floating points are saved as double and are transformed in fraction number just before any processing. For the bus relation, *P* and *Q* are supposed to be integer and not double.

Between two iterations, an error is obtained for each complex network node. The modulus obtained is then calculated, and the largest value among all values is considered the current value to be used as convergence criterion in relation to a predefined threshold ε. Note that, this method is better and more rigorous, which calculates the error at one point in the network, based only on either the real parts or the imaginary parts. The structure of the entities in Figure 1 is straightforward. The “busType.id” identifies the type of a bus (Slack, PV …), “netword.id” identifies a network in the database since it can save data of many networks. The entity “bus” contains all the buses in the system and the attribute “busType.id” captures the type of the bus. “Algorithms” contains all the algorithms that can be simulated (Gauss-Seidel, NR, FD ...) while “iterations” contain rows of the result of simulations that help compare the algorithms implemented with double and with fractional numbers.

IV. Description of the Platform

The main architecture of the platform is shown in Figure 2. We can say that, the learning process is simple and is similar to the process described in [7]. Telemetry data allows building of new topologies and agents to compare current situations with old ones. If they are different, the new situation is stored. If the situation exists, the configuration is read from the database and/or a proper diagnostic is done in comparison to a similar case that occurred in the past, and that has been confirmed by an expert. This configuration has been previously validated and entered into the database by the expert. The methodology for designing multi-agent used is Prometheus [9] which according to us is one of the simplest methodologies despite the fact that, the generation of the code in Jack is done via a commercial tool.

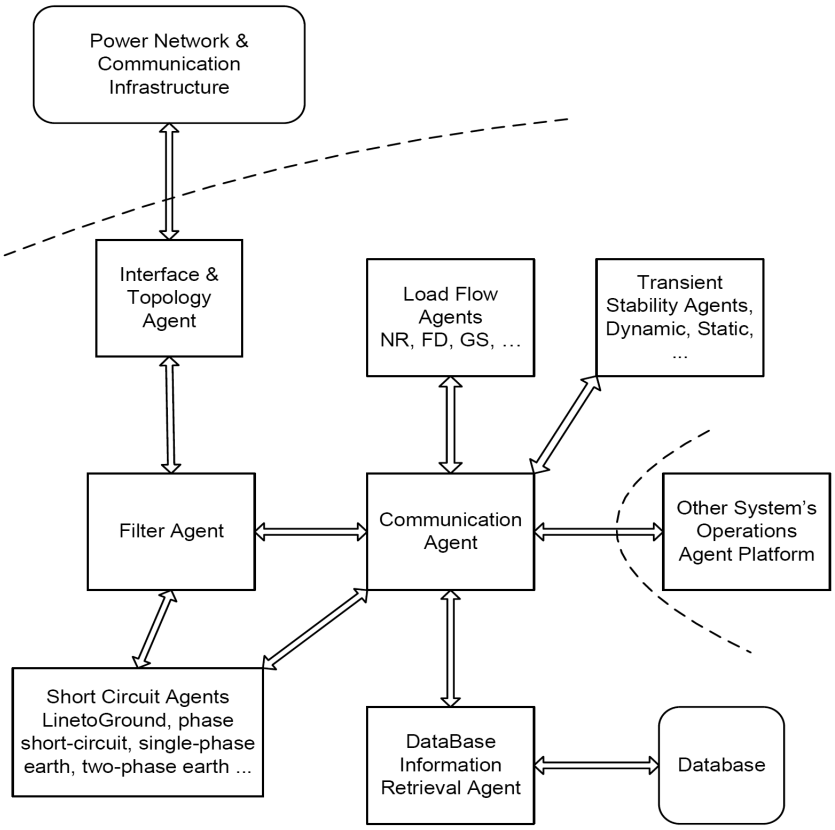

Figure 2. Multi-agent platform for electric power network analysis operations.

A. Roles of main agents of the platform

Figure 2 shows the main agents of the platform. Their roles are described as follows:

- **Interface & Topology Agent (ITA):** It is connected to the physical system and its role is, to collect telemetry measurements, form a logical topology of the network and transmit data to the filter agent. If measures are not read in a predefined interval, it triggers an alarm that communicates to the filter agent, which will transfer to the diagnostic platform not shown here via the communication agent. In the `OnStart method()`, a simple behaviour `One-`

`ShotBehaviour()` is executed and then another type behaviour `onTickBehaviour()` is executed in a configurable interval of 15000.

- **Filter Agent (FA):** At initialization, it is the parameters describing the different topologies of the network requested from the database via the communications agent. Topologies sent by the ITA are constantly compared, and new topologies are stored in the DB. With each new topology, warning signals are sent to other agents to re-adjust and make new calculations.
- **Load Flow Agents (LFA):** There are several agents that calculate the LFP each with its own algorithm. It is a holonic structure that must choose the best algorithm and store it in terms of computing time, accuracy, according to the current topology. Of course, if the current topology is identical to a previous topology, the LFP is no longer calculated.
- **Short-Circuit Agents (SCA):** Like the LFA, it is considered as holonic. It includes four micro agents for calculating short-circuits. The different types of short-circuits: three-phase, single phase-earth, two-phase isolated, two-phase-earth. The results are sent to the CA for storage in the database via the interface agent. If similar situation is detected, the FA can prevent storage. If there is a new case, then the data is stored in the DB.
- **Transient Stability Agent (TSA):** This agent is in charge of the stability of the network. In view of the complexity, it is also regarded as a holonic. The process of storing results of the stability is decided by the FA.
- **Communication Agent (CA):** This agent is considered as a router responsible for routing communications to different agents. If in doubt, it consults with the FA to decide or not to route a call.
- **Data Base Information Retrieval Agent (DBIRA):** This is an agent to interface with the database. It receives SQL requests from the CA,

reformats as needed and sends to the RDBMS used. It executes the query and returns the result that is responsible for communicating to the sender of the request through the CA.

B. Some behaviours of agents of the platform

The platform may be connected to another platform such as diagnosis, planning or restoration of the network. Such a platform has already been described for transformers' diagnostic of using dissolved gas analysis [7].

The Jade platform was accessed from a Java application, developed with Eclipse via the library that implements the pattern `jade.core.Runtime Singleton`. The Singleton Runtime provides methods `CreateMainContainer()` and `CreateAgentContainer()` where agents are created.

The ITA behaviour `Initialize()` with the prototype `public class Initialize {} extends One-ShotBehaviour` helps repatriate data that will allow the FA to build the network topology. The method `DispatchInstructions () extends generic` behaviour while the method `MonitorPNW ()` extends the behaviour `TickerBehaviour ()` adjustable according to the telecommunication infrastructure associated with the network. The events are traditional events for multi-agent systems [3].

Most of the other agents are notified to launch their different behaviours. For this, we used the class `jade.core.BaseService`. The short-circuit processing, for example, can be started by calling the service `public class extends ShortCircuitService Base-Service {}`.

C. Development approaches and tools

The system is being implemented in two variants. The first one which is a variant accessing the Jade environment as described in Subsection IV.C, and the second one with enterprise web technology, which is an undergoing work. This second variant is done using the Eclipse platform Helios JavaEE patched with `EclipseLink, DaliJPA` and MySQL RDBMS.

Middleware is JBoss and the connector to the platform is `Jademx` Jade. The server is `Tomcatservlet` container. Some codes were generated with `CIMTOOL` and others with the generator developed in [13] that comply with the Model View Controller.

Table 1. Comparison of some implemented algorithms with double and fractional numbers

Network	Number of nodes	Number of iterations for the classical NR	Number of iterations for the classical Jacobi	Number of iterations for NR with fractions	Number of iterations for Jacobi with fractions
Lynn	5	4/4	29	4/4	23
RISC	25	9/9	32	9/9	35

V. Simulation Results and Diagnostic

The fractional numbers allowed having more précised results. For example, the last element of the admittance matrix in [15] `12.3570117 – j29.48`5605 and is `12.357012 – j29.51`560514 in our software. Further calculations bring more different results. A diagnosis for the separation of the network can be done by analysis of the topology matrix, and the LFP computed is stored under normal conditions. If the network has abnormal impedance or abnormal voltages, a module concludes on the detection of an abnormality in the network. For the LFP, Table 1 presents a comparison of two algorithms implemented probably with double types with ours actually implemented with fractional numbers. The 25 nodes of the RISC representing the interconnected South Cameroon network and the network of 5 nodes have been tested. We voluntarily removed the lines data were not available and fields were empty in Table 1. The application is under test on the IEEE118 bus.

VI. Conclusion and Future Work

Agent oriented programming is a programming paradigm suitable for

enterprise applications in electrical engineering. In this paper, we have demonstrated the feasibility of using fractional numbers to improve accuracy and accelerate the calculations in the algorithms of the LFP. A platform was presented for the analysis, particularly the LFP and the calculation of short-circuit. The work is outstanding in terms of evolution, is the section on transient stability. It would be interesting to find several other networks with known numbers of iterations, to be compared with results of our application to validate the approach for cases that slowly converge. The system is actually used as simulation software and should be tested in order to validate the learning process in the multi-agent part.

References

[1] H. A. Amir, B. Kaustav and P. Massoud, Scaling analysis of on-chip power grid voltage variations in nanometer scale ULSI available at http://atrak.usc.edu/~massoud/Papers/powergrid-voltagedrop-journal.pdf

[2] I. S. Baxevanos and D. P. Labridis, Implementing multiagent systems technology for power distribution network control and protection management, IEEE Trans. Power Delivery 22(1) (2007), 433-443.

[3] F. Bellifemine, G. Caire and D. Greenwood, Developing Multi-agent Systems with JADE, John Wiley and Sons Ltd., 2007.

[4] Chung-Wen Ho, A. Ruehli and P. Brennan, The modified nodal approach to network analysis, IEEE Trans. Circuits and Systems 22(6) (1975), 504-509.

[5] J. K. Kok, C. J. Warmer and I. G. Kamphuis, Power matcher: multi agent control in the electricity infrastructure available at http://vtb.engr.sc.edu/vtbwebsite/downloads/publications/ISAP_agent_Ferdi.pdf

[6] P. Kundur, Power System Stability and Control EPRI, McGraw Hill Inc., 1993.

[7] J. P. Lienou, M. Nkenlifack, E. Tanyi and T. Noulamo, A generic multi agent-based platform for reliable diagnostic by DGA, Adv. Computer Sci. Eng. 5(1) (2010), 11-23.

[8] R. J. Marti, L. R. Linares, J. A. Hollman and H. A. Moreira, OVNI: integrated software/hardware solution for real-time simulation of large power systems, Conference Proceedings of the 14th Power System Computation Conference PSCC02, Sevilla, Spain, June 24-28, 2002, available at http://www.ece.ubc.ca/~jrms/02-06_OVNI_MATE.pdf

[9] O. Maslennikow, V. Lepeha and A. Sergyienko, FPGA implementation of the conjugate gradient method, 6th International Conference, PPAM 2005, Poznań, Poland, September 11-14, 2005.

[10] N. Maslennikowa, O. Maslennikow, R. Berezowski and J.-P. Lienou, Design of Fpga-based multi-operand modular adders for residue number system converters, IEEE Proceedings of the International Conference Mixed Design of Integrated Circuits and System, 2006.

[11] S. D. J. McArthur, E. M. Davidson, V. M. Catterson, A. L. Dimeas, N. D. Hatziargyriou, F. Ponci and T. Funabashi, Multi-agent systems for power engineering applications - Part I: Concepts, approaches, and technical challenges, IEEE Trans. Power Systems 22(4) (2007), 1743-1752.

[12] P. H. Nguyen, W. L. Kling, G. Georgiadis, M. Papatriantafilou, L. A. Tuan and L. Bertling, Distributed routing algorithms to manage power flow in agent-based active distribution network, 2010 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT Europe), Oct. 11-13, 2010, pp. 1-7.

[13] T. Noulamo, E. Tanyi, M. Nkenlifack and J. P. Lienou, Domain specific model and generic architectures for control and monitoring of dynamic systems, Adv. Computer Sci. Eng. 4(1) (2010), 55-75.

[14] L. Padgham and M. Winikoff, Developing Intelligent Agent Systems: A Practical Guide, Wiley, 2005.

[15] Lynn Powell, Power System Load Flow Analysis, McGraw Hill Professional Engineering, 2005.

[16] S. Rumley, E. Kägi, H. Rudnick and A. Germond, Multi-agent approach to electrical distribution networks control annual, IEEE International Computer Software and Applications Conference, available at http://web.ing.puc.cl/~power/paperspdf/RumleyCompsac.pdf