Tomasz CHMIELEWSKI[1], Piotr MARS[1], Miłosz MIŚKIEWICZ[1], Paweł BŁASZCZYK[1]

ABB Corporate Research Center in Krakow (1)

doi:10.15199/48.2017.09.15

Universal DLL based components for simulations of multiphysical electro-thermal systems

Abstract. *This paper presents the procedures for universal models preparation of components and devices that can be used in various simulation software environments that can be used for electro-thermal systems analysis. Such approach is applicable for multiphysical analyses in areas such as electrical engineering, electronics and physics. The proposed method that involves generation of DLL libraries is explained based on multiphysical model comprising electric and thermal subsystems. The paper highlights a complementary benefit of models black-boxing that may play an important role for know-how protection. The analyses performed herein revealed that proposed universal models are able to produce credible quantitative results in case of all considered software environments (PSCAD, Matlab, DIgSILENT), however with significant differences in the recorded computation time.*

Streszczenie. *Niniejszy artykuł przestawia metodę przygotowania uniwersalnych modeli komponentów, możliwych do użycia w różnych środowiskach symulacyjnych służących do analiz multifizyczyny w dziedzinach takich jak elektrotechnika, elektronika i fizyka. Zaproponowana metodyka uwzględniająca generację bibliotek DLL została przedstawiona w oparciu o multifizyczny model składający się z części termicznej i elektrycznej. Przeprowadzona analiza wykazała dużą zbieżność wyników pomiędzy rozpatrywanymi programami symulacyjnymi (PSCAD, Matlab, DIgSILENT), jednakże z zauważalnymi różnicami w długości trwania obliczeń.* ***Uniwersalne modele komponentów bazujące na bibliotekach DLL do symulacji multifizycznych systemów elektro-termicznych***

Keywords: PSCAD, DIgSILENT, Matlab, multiphysics, simulation, modelling
Słowa kluczowe: PSCAD, DIgSILENT, Matlab, symulacje multifizyczne, modelowanie

Introduction

Nowadays computer simulations are an indispensable aid tool for design, development and research. Thanks to variety of modern simulation software environments nearly every physical phenomenon can be digitally reproduced and accordingly analysed. Such approach is a cost efficient solution for design optimization, non-destructive testing or product development. As can be seen in Fig. 1 which exhibits nowadays common approach to research process, the simulations (virtual experiment) are typically a step preceding the laboratory experiments. However, if model has been already verified experimentally the simulation may as well become nearly fully credible substitute for experiments.

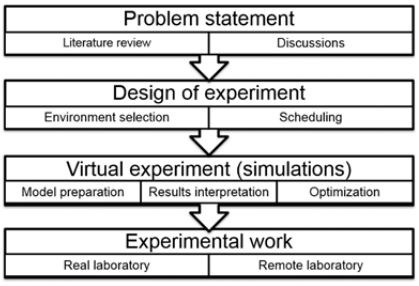


Fig. 1. Modern research approach

Despite the rapid development of computation technology and decreasing impact of hardware constraints, certain limitations of the simulation software still exist. This fact stimulates the development of highly specialized software packages, dedicated for small range of phenomena. Hence the software vary significantly in their functionalities, elements libraries, user interfaces, performance rates etc. Due to the increasing number of cross-area projects there is often a need for know-how and model exchange. Such demand is sometimes problematic to be met due to significant differences between the simulation environments. Therefore, it is beneficial to create procedures for model or elements import/export in order to accelerate the analysis (the time consuming model preparation could be possibly avoided). Another motivation for adopting the models from the foreign software is possibility of increasing the performance of simulations. This feature could for instance be of a particular importance in case of software that provides parallel threads calculations. The possibility of software capability (universality) improvement is also a great benefit of flexible model exchange. For example, such procedure could potentially facilitate multiphysical simulations that involve combined e.g. electrical and mechanical elements.

There are essentially three ways that allow to adapt whole or partial model to the external software (Fig. 2), however most of them requires a substantial effort for the model conversion. The most time consuming is the straightforward method i.e. building the model from scratch followed by the verification with use of base model. However some of the programs provide automated modules that allows the quick import of models from external programs (e.g. PSCAD, NEPLAN, DIgSILENT). It is usually very limited and non-flexible functionality. The indirect way to combine models from two software environments is extraction of data from one software and feeding it as an input to another one. Moreover, in some cases there is a default online link that allows simultaneous simulations in two different programs (e.g. PSCAD and Simulink).

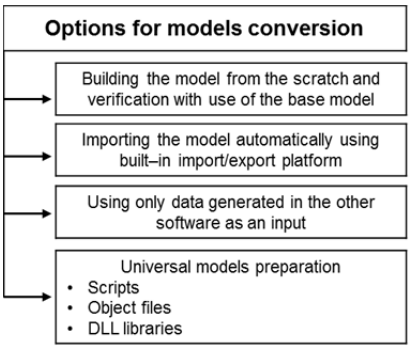


Fig. 2. Possibilities for including the foreign models from external software

The method presented herein which is based on DLL libraries utilisation can be considered as one of the most efficient and universal. What is important, the utilization of DLL libraries features a black-boxing of the model (at least to a certain degree). The benefits of combining the different software can be also found in [1–10].

This paper aims mainly at comparison of different simulation software environments that use a DLL based primary model. In such a way it can be clearly demonstrated how the calculations time varies depending on the software applied.

Universal DLL models preparation procedure

In order to generate the multi-system simulation model, an universal tool must be used. It can be observed that much more companies are supporting external model interfacing based on the Dynamic Linked Library objects (DLLs) [11–13]. The library can be prepared in any environment (windows-based) and simulated in different simulation software.

The key issue is to provide a clear interface between DLL and simulation software. One promising mechanism is an IEC standard [14] which defines what functions should the DLL include and how they are called. Main functions are:

- Model_Getinfo() providing general information about the model,
- Model_Instance() creating the instance of the model (multi-instancing is supported),
- Model_Initialize() initializing the model,
- Model_Outputs() performing simulation time step giving model outputs,
- Model_Terminate() shutting down a specified model instance.

As an example, DLL generation process from Simulink model has been presented. The process can be divided in three main stages:

1. Creation of simulation model including DLL-to-be block and test harness,
2. Generation of DLL from the block,
3. Testing DLL in the root-software (Matlab/Simulink).

Creation of simulation model

In the first step simulation model including test harness should be created. It is crucial because testing DLL in the 3rd party software can be problematic. For example in case of bug, it is hard to be located, especially if it is related to the implemented algorithm, interface or to the simulation software which is used. An example test system is presented in Fig. 3. Yellow block is to-be-DLL system and grey one is a placeholder for DLL. There are two main reasons to have that system: First, check the system behaviour. Second, check (in the step 3) if DLL works in the same way as the base system. In case of this paper the original model was created in Matlab/Simulink software.

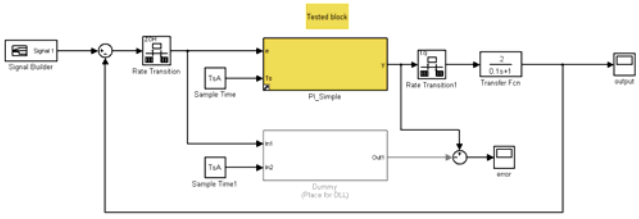


Fig. 3. Test harness

Generation of DLL

DLL in Matlab/Simulink can be generated automatically from the source model using code generation tool (what requires Matlab, Simulink and Embedded Coder ® toolboxes).

The output of the code generation is a set of source (*.c) and header (*.h) files which can be used in an external project or be compiled through Matlab. For that, the compiler must be associated and makefile should be generated. The makefile contains instructions how to create the project, how to compile it and what files should be generated (Fig. 4). In case of this paper the resulting DLL model is compliant with the IEC 61400-27-1 standard, Annex F (Generic Software Interface for use of models in different software environments) [14]

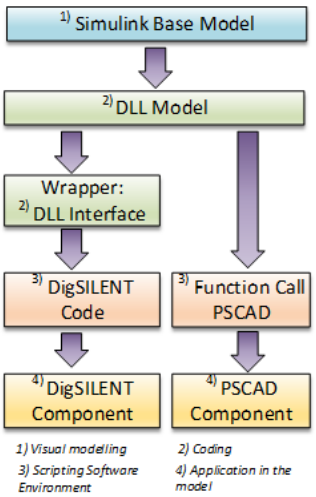


Fig. 4. DLL generation process

DLL testing

The final step is DLL testing which can be performed in the test harness presented in Fig. 3. However, first the Simulink block calling DLL should be created. The process of block generation is a straightforward – it is required to call a few-line script which generates a block calling defined DLL. Next the block can be placed in the place of grey block in Fig. 3 and output of its block should be compared with an original output (of the yellow block on the Fig. 3). If the error is on the acceptable level (some rounding errors can be observed), DLL can be distributed. If no, generated code must be checked to find the root cause of the difference between the base model and DLL.

Description of utilized simulation environments

DIgSILENT stands for DIgital SImuLation of Electrical Networks. This simulation environment is a CAE tool for the modelling and analysis of transmission, distribution, and industrial electrical power systems. The software is designed as an advanced integrated and interactive simulation software tool package dedicated to electrical power system and control analysis in order to reach the main objectives of planning and operation optimization.

The IEC compliant dynamic heater library for EMT and RMS simulations is implemented in PowerFactory using the “digexfun interface”. The functionality of the “C – Interface” is implemented in the DSL (DIgSILENT simulation language) functions that are provided by the “digexfun interface” (via the compiled DLL). The digexfun.dll file acts as a middle layer that calls the functions of the heater library and makes the results available to PowerFactory

simulation SW. This solution supports multiple instances of one external DLL model [11].

Nonlinear equations are solved using iterative method like the Newton-Raphson algorithm." For heavily loaded large transmission systems, the classical Newton-Raphson algorithm utilizing the Power Equations formulation generally converges best. Distribution systems in particular unbalanced distribution systems are generally better coverage utilizing the Current Equations formulation [11].

The PSCAD program (Power System Control Aided Design) is used for modelling and simulation of various types of power systems. It is most commonly used to simulate the time-domain transients in power systems. It allows modelling them with taking into consideration measurement and control systems.

The DLL model implementation in PSCAD environment is easy and no additional DLLs has to be created. However for supporting multiple instances it is needed to attach separate DLL file for each instance [12]

Simulink is an environment which uses block diagrams to create complex systems for multiphysical simulations. It is dedicated for model-based design approach. Furthermore it supports simulation, and automatic code generation which has been used to create the code script from block diagram heater model and then to generate the DLL model. [13]

Case study description

The test system used for universal model procedure presentation and testing comprises thermal and electrical subsystems. The room temperature control system has been modelled comprising the room thermal model – [5], the electric radiator and the PI controller - Fig. 5.

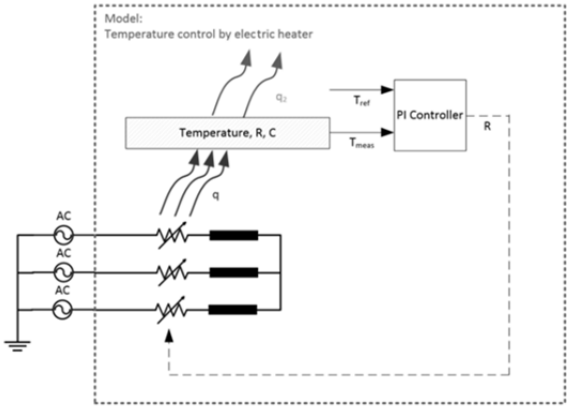


Fig. 5. Room temperature control system – overall diagram

The ambient temperature is considered constant. The heat q_i used as the input for thermal model is generated by phase currents flowing through resistors in accordance with the Joule - Lenz law presented in equation [15] (1).

$$q_i = 3R_{ph} i_{ph}^2 = 3R_{ph}\left(\int\left(\frac{u_S - u_R}{L}\right)dt\right)^2 \quad (1)$$

The current is expressed as the integral of the voltage drop across the inductor over an inductance. This results from the transformation of the equation (2) describing relation between voltage and current across an inductor.

$$u_L = u_S - u_R = L\frac{di}{dt} \quad (2)$$

where: q_i – heat, u_S – supply voltage, u_R – voltage drop over resistance, u_L – voltage drop over inductor L – heater inductance, R_{ph} – phase resistance of the heater, i_{ph} – phase current .

The heater thermal model cooperation with the surroundings is described by the (3).

$$q_i = \frac{T_{meas} - T_{ref}}{R_{ra}} + C_r\frac{dT_{meas}}{dt} \quad (3)$$

The equation (3) can be rewritten to the form presented in (4).

$$\frac{dT_{meas}}{dt} + \frac{1}{C_r R_{ra}} T_{meas} = \frac{q_i}{C_r} + \frac{T_{ref}}{C_r R_{ra}} \quad (4)$$

The solution of (4) determines the room temperature which states the model output and is compared to the reference value – eq. (5).

$$T_{meas}(t) = q_i(t)R_{ra} + e^{-\frac{t}{C_r R_{ra}}}\left(\frac{T_{ref}}{C_r R_{ra}}t - q_o R_{ra}\right) \quad (5)$$

where: q_i – heat function of time, q_o – Initial heat T_{meas} – measured room temperature, T_{ref} – reference ambient temperature, R_{ra} – resistance between room and ambient, C_r – room thermal capacitance.

The difference between measured and reference temperature yields the error value for the input of PI regulator which controls the phase currents by changing the electrical circuit phase resistances. As a result, PI regulator controls the amount of the generated heat thus contributing to increase of the room temperature. In the DLL model approach which is depicted on the Fig. 6 electric circuit is represented as the controlled current source.

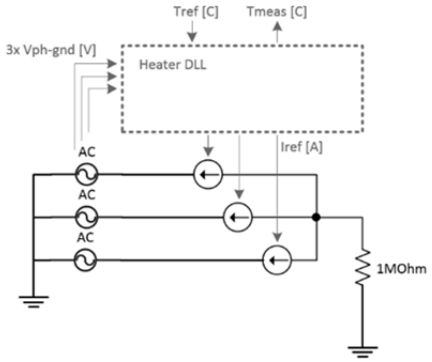


Fig. 6. Test model with implemented DLL based element

Phase voltages and the reference temperature value are the inputs of the DLL model – Table 1.

Table 1. DLL model inputs

Name	Unit	Description
V1	[V]	Phase 1 to ground voltage measurement
V2	[V]	Phase 2 to ground voltage measurement
V3	[V]	Phase 3 to ground voltage measurement
Tref	[°C]	Temperature reference

Input signals interact with the hidden inside the DLL mathematical model which represent the electrical heater and its cooperation with the environment. After processing input signals the DLL model returns the reference phase current values for controlling the current source and the measured room temperature – Table 2.

Table 2. DLL model outputs

Name	Unit	Description
I1ref	[A]	Phase 1 current reference
I2ref	[A]	Phase 2 current reference
I3ref	[A]	Phase 3 current reference
Tmeas	[°C]	Temperature reference

Table 3. Model parameters

Parameter	Value	Description
Ts	0.1e-3	Sample time [s]
L	0.01e-3	Heater inductance [L]
C_r	100e3	Room thermal capacitance
R_{ra}	1e-6	Room-ambient resistance

The values of model parameters used in simulations are listed in the Table 3.

To compare calculation time of the DLL model in various simulation environments, PSCAD, Matlab/Simulink and DIgSILENT PF software have been installed on the same computer with parameters presented in Table 4. Simulation software versions are given in the Table 5.

Table 4. Parameters of computer used for simulations

System	Windows 7 Enterprise
Processor	Intel® Core™ i7-3720QM CPU @ 2.60 GHz
RAM	12.0 GB
System type	64-bit

Table 5. Software version used for simulations

Software	Version
PSCAD	4.6 withGFortran 4.2.1
Matlab	7.9 (2009b) withSimulink 7.4, SimPowerSystems 5.2
DIgSILENT	PowerFactory 15.2.7 64-bit

For comparing the simulation performance time the classical test case was created. Namely, the reference temperature STEP signal from 10°C to 22°C was supplied to the input of DLL model. These conditions were the same for each of instances. Simulation time has been measured for every test case. For better performance all measurements visualisations like current, temperatures, voltages, etc. were switched off.

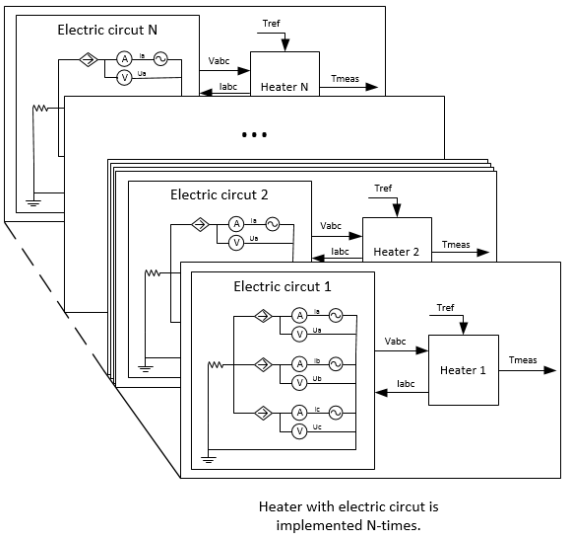


Fig. 7. Multi-instancing modelling approach

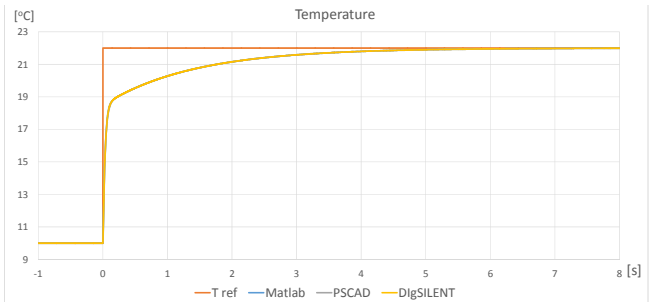
The simulations were performed for different number of simultaneous instances: one, two, four, six, eight and ten. One instance means that one room temperature control system has been used during simulation running, this approach is visualized in the Fig. 7

Measurements of the simulation execution time was done by using tic toc function in Matlab/Simulink, observing runtime messages (total CPU time parameter) in PSCAD and for DIgSILENT the Python script was used which runs the simulation, stops it and returns the simulation execution time value. Every simulation has been performed ten times and average value of execution time has been calculated.

Results and discussion

Model has been interfaced in three different simulation environments as the universal DLL model. Convergence test employing one control system has been initially performed. Temperature measurement has been taken as it is presented in the Fig. 8a.

a)



b)

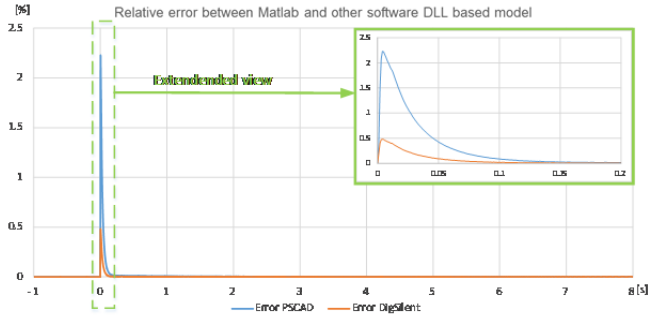
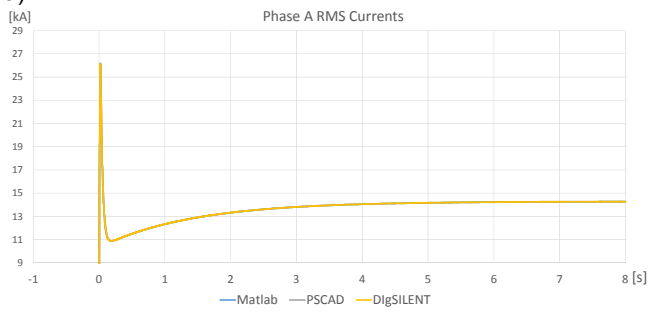


Fig. 8 Temperature response for reference step change: (a) – temperature trace, (b) – relative error between the Matlab/Simulink DLL model and other utilized software

RMS value has of one phase current has been also shown for the same purpose (Fig. 9a).

a)



b)

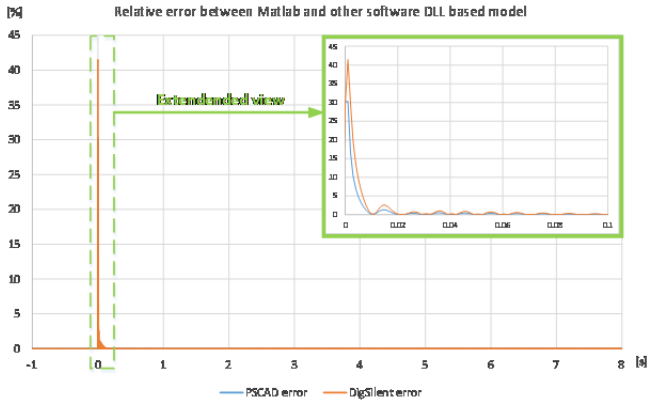


Fig. 9. Step response – RMS value of current phase A: (a) – current trace, (b) – relative error between Matlab/Simulink DLL model and other utilized software

Both, temperature and current curves are showing good convergence between Matlab, PSCAD and DIgSILENT PowerFactory. The relative error calculated for both parameters is time dependent. In Fig.8b and Fig.9b it can be observed, that the highest difference between the base Matlab/Simulink model occurs at the first steps of the simulation. The error exhibits a very fast decay for both temperature and current and reaches values close to zero after no more than 0.2 seconds of the simulations.
The graph of execution times for different simulation environments is shown in the Fig. 10.

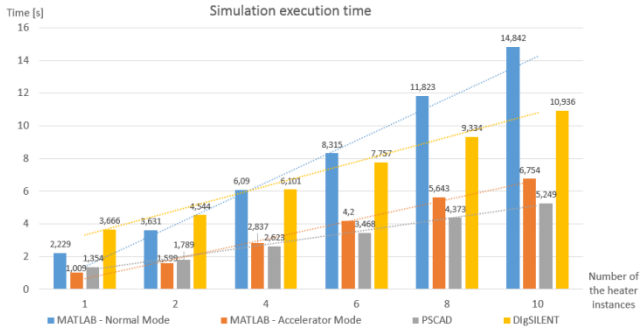


Fig. 10. Time of simulation runs for different number of instances in model

As can be observed, PSCAD gives the best performance when four to ten instances in model are introduced. Execution time of model comprising one or two control systems is shortest in Matlab/Simulink operating in accelerator simulation mode. The slowest environment is recognized to be Matlab/Simulink at normal operation mode and for model containing six and more instances and DIgSILENT PowerFactory for lower number of instances.

Conclusions

The procedure for universal DLL based model preparation presented in this paper demonstrated a few interesting and potentially beneficial features. One of the greatest advantage of the DLL modelling approach is the protection of intellectual property. DLL models act as a black box, details about the system are not visible for users. Other advantage of using DLL models is fast implementation in various simulation software. The once performed framework can be modified and used for other DLL. Furthermore this approach enables the easy multiple instances implementation in various simulation software, which may effectively decrease the simulation execution time. Furthermore, utilization of the DLL based model produces the error on the acceptable level with only some rounding errors visible. All in all, the DLL based universal model may be extremely useful both in terms of the technical features and the business operation.

***Authors**: mgr inż. Tomasz Chmielewski, mgr inż. Piotr Mars, mgr inż. Miłosz Miśkiewicz, mgr inż. Paweł Błaszczyk, ABB Corporate Research Center, ul. Starowiślna 13a, 31-038 Kraków, E-mail: tomasz.chmielewski@pl.abb.com, piotr.mars@pl.abb.com, milosz.miskiewicz@pl.abb.com. pawel.blaszczyk@pl.abb.com,*

REFERENCES

[1] S.H.A. Niaki, H.K. Karegar, M.G. Monfared, Electrical Power and Energy Systems A novel fault detection method for VSC-HVDC transmission system of offshore wind farm, Int. J. Electr. Power Energy Syst. 73 (2015) 475–483. doi:10.1016/j.ijepes.2015.04.014.
[2] X. Wang, M. Yue, E. Muljadi, PV generation enhancement with a virtual inertia emulator to provide inertial response to the grid. Energy Conversion Congress and Exposition (ECCE), (2014), 17-23.
[3] F. Shahnia, A. Ghosh, G. Ledwich, F. Zare, Electrical Power and Energy Systems Voltage unbalance improvement in low voltage residential feeders with rooftop PVs using custom power devices, Int. J. Electr. Power Energy Syst. 55 (2014) 362–377. doi:10.1016/j.ijepes.2013.09.018.
[4] L. Davila-gomez, A. Colmenar-santos, M. Tawfik, M. Castro-gil, Simulation Modelling Practice and Theory An accurate model for simulating energetic behavior of PV grid connected inverters, Simul. Model. Pract. Theory. 49 (2014), 57–72. doi:10.1016/j.simpat.2014.08.001.
[5] T.S. Abuaisha, General study of the control principles and dynamic fault behaviour of variable-speed wind turbine and wind farm generic models, Renew. Energy. 68 (2014) 245–254. doi:10.1016/j.renene.2014.01.004
[6] Souli, Aissa, Abdelhafid Hellal, and Slami Saadi. Programming EMTP-ATP-PSCAD Software Functions using MATLAB for Power Systems Transient Analysis. Przegląd Elektrotechniczny 86.6 (2010): 285-287.
[7] Marciniak, L. Implementacje modeli łuku ziemnozwarciowego w programach PSCAD i Matlab/Simulink. Przegląd Elektrotechniczny 88.9a (2012): 126-129..
[8] Liu, Xiaolei, A. H. Osman, and O. P. Malik, Advanced simulation tool for relay testing. Power Symposium, 2008. NAPS'08. 40th North American. IEEE, (2008).
[9] P. Bjorklund, J. Pan, C. Yue, K. Srivastava, A New Approach for Modeling Complex Power System Components in Different Simulation Tools, 16th Power Syst. Comput. Conf. (PSCC 2008 Glas. 1 (2008).
[10] Troudi, M., et al., Macro-modeling for the compact simulation of single electron transistor using SIMPLORER., Microelectronics Journal 38.12, (2007), 1156-1160.
[11] DIgSILENT GmbH, DIgSILENT PowerFactory User Manual, (2015) 453–454, 633–646.
[12] PSCAD User Manual, (n.d.).
[13] Matlab/Simulink User Manual, (n.d.).
[14] IEC 61400-27-1, Electrical simulation models – wind turbines, edition 1.0, (2015) 81–86.
[15] W.M. Rohsenow, J.P. Hartnett, Y.I. Cho, Handbook of heat transfer, McGraw-Hill New York, (1998).