ARCHIVES OF ELECTRICAL ENGINEERING VOL. 59(3-4), pp. 113-120 (2010)

DOI 10.2478/s10171-010-0009-9

Squirrel-cage motor drive dynamics modeling including chosen damages – programming tools

ADAM SOŁBUT

Department of Power Electronics & Electrical Drives
Bialystok University of Technology

e-mail: a.solbut@pb.edu.pl

(Received: 24.06.2010, revised: 24.08.2010)

Abstract: Description of program tools simplifying simulation applications building for physical phenomenons described by differential equations in state equations form modeling is presented in the paper. A method for using prepared libraries for squirrel-cage motors including any motor damages modeling had been described. For that purpose, squirrel-cage motor mathematical model in natural coordinates system had been presented. Presented solutions provide also supply sources (inverters) modeling, including their microprocessor implementation and other phenomenons, that assume state equation structure step changes, depending on variable limitations and time value.

Key words: numerical modeling, asynchronous motors, diagnostics, object oriented programing tools

1. Introduction

Industrial inverter solutions provide many different control algorithms. Algorithms related with control structures for asynchronous, synchronous and DC motors with electronic commutation have dominating meaning. Practically, permissible current value limiting procedures and controller auto-tuning procedures are also used. Issues of motors diagnostics ale usually ignored. Only the cases of damages that prevent further motor run are included. Searching for squirrel-cage motor state estimation algorithms during their normal exploitation requires creation of program tools, that provide motor's behavior modeling in any damage states, using any control method. Available papers provide many different modeling tools for many different damage states [2-4, 7]. Depending on the type of damage, different squirrel-cage motor's mathematical models may be used.

Continuous microprocessors development is followed by significant changes in their programming tools. Methods of object oriented reality description are the essence of modern programming languages [1, 5, 9]. The most often used for microprocessor programming C language had been expanded to object-oriented programming tools (C++). Present-day programming techniques became a powerful tool providing any complex technical issues

modeling. Libraries supporting differential equations numerical solving, mechanisms providing microprocessor-realized algorithms testing and tools facilitating asynchronous motor modeling in natural coordinates configuration are presented in the paper. Presented tools will be used for inverter-fed squirrel-cage asynchronous motors modeling, providing any motor damages and different control algorithms.

2. Library supporting non-linear differential equations solving

Proposed simulation applications building mechanism depends on separation of the numerical integrating procedures from mathematical description of the process. The modeled process is described in *UserProces* class (Fig. 1).

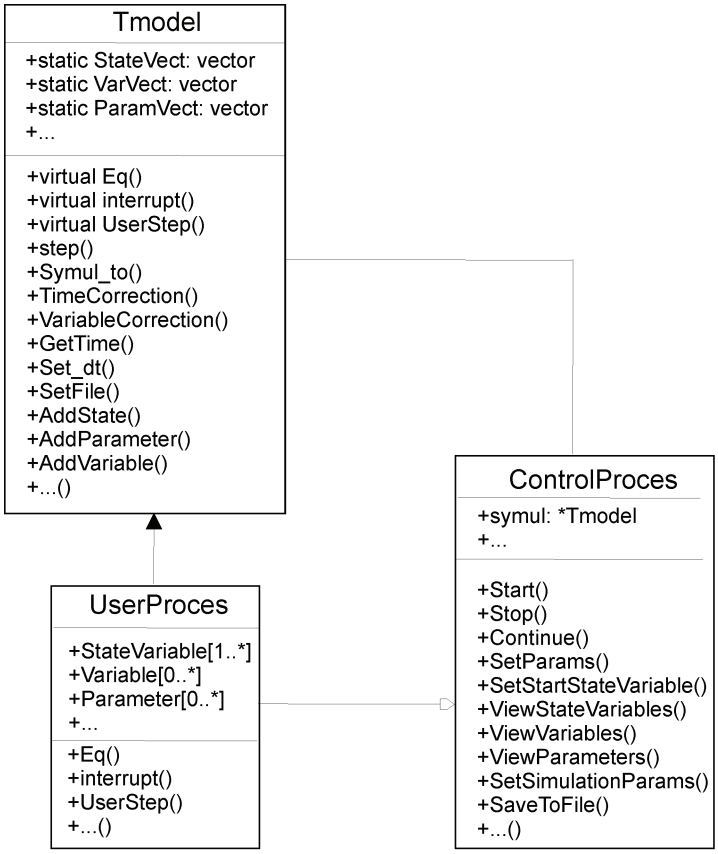

Fig. 1. Simplified simulation applications class diagram using proposed program solutions

The elementary class for *UserProces* is the *T-model* class, in which numerical integration procedures are contained. State equations integrating process is realized through the *symul_to()*

method. After *symul_to( )* activation, it realizes static table *StateVect* objects integration process. The table contains *StateVariable* type objects pointers.

Objects of this type include actual state variable value, name and other variables required for numerical integration process. Control process mechanisms are placed in the *Control-Proces* class.

Simulation application building is based on creation of one class containing process description. State equations notation, state variables, supplementary variables and parameters declaration is necessary (in the *Eq* method). Simplicity of state equations notation had been obtained by specific structure of the *StateVariable*, *Variable* and *Parameter* object types constructors and numerical operators overloading.

UserProces class can be placed in a DLL library, and *ProcesControl* class is an application that uses this library. Proposed solution is a handy tool, in which limitations of the process description are dependent on the C++ language capabilities. The only limitations for using it are the necessity of using *StateVariable* type objects for storing information about state variables values and the notation of state equations in the Eq method.

Other mechanisms placed in the *T-model* class provide for example:

- possibility of control process modeling, using microprocessor technology (interrupt method);
- possibility of those processes modeling, in which changes in state equations structure in any time moment (*TimeCorrection* method) or programmed state variable values (*VariableCorrection* method) are acceptable.

Those mechanisms were intentionally created for motor drives with asynchronous motors fed by voltage inverters modeling.

Variable and *Parameter* object types were created for simulation tests simplification, after the application is compiled. Compiled application provides for example:

- starting, stopping and continuing simulation,
- parameters values changing during the simulation,
- start state variable values changing,
- state variables and supplementary variables values changes observing,
- writing of the numerical modeling results to files in acceptable formats for the *Anagraf_win* program (for analysis and graphical presentation of test results).

In complicated applications, it's often needed to use a code, that's executed after each and every one correctly ended simulation step. For that goal, a virtual method *user_step* had been prepared. It returns a value of true in case of neccesity of repeating calculations in the last integration step.

Depending on the library version, *T-model* class may include a Windows core objects table, which is needed for multithreads interactive application building in Win32 environment. Described libraries are available in http://we.pb.edu.pl/~solbut/zip/symul_dll.zip as an example source code.

3. Numerical model of an asynchronous motor

Mathematical model of a symmetrical asynchronous motor had been shown in many literature positions [3, 4, 6]. Motor models including internal asymmetries are usually based

on the transformations that include air gap magnetic field higher spatial harmonic components. Owing to quick computer techniques progress, practical use of models based on the equations in the natural coordinates system is now possible, where influence of damage states is included in dependencies describing model resistance and inductance matrixes.

Equations of asynchronous motor with m_1 stator and m_2 rotor phases can be described by vector equations:

$$[U] = [R][I] + \frac{d}{dt}[\Phi], \tag{1}$$

$$J\frac{d\Omega}{dt} = T_e - f\,\Omega - T_0\,, \tag{2}$$

where:

$$[R] = \begin{bmatrix} [R_s] & [0] \\ [0] & [R_r] \end{bmatrix}, \tag{3}$$

$$[\Phi] = \begin{bmatrix} [\Phi_s] \\ [\Phi_r] \end{bmatrix} = [L][I] \begin{bmatrix} [L_s] & [M_{sr}] \\ [M_{rs}] & [L_r] \end{bmatrix} \begin{bmatrix} [I_s] \\ [I_r] \end{bmatrix}, \tag{4}$$

$$[U] = \begin{bmatrix} [U_s] \\ [U_r] \end{bmatrix}, \tag{5}$$

$$[I] = \begin{bmatrix} [I_s] \\ [I_r] \end{bmatrix}, \tag{6}$$

$$T_e = \frac{1}{2}[I]^T \left\{ \frac{d}{d\Theta}[L] \right\} [I], \tag{7}$$

$[R]$ – resistance matrix; $[\Phi]$ – fluxes matrix; $[L]$ – inductance matrix; $[U]$ – supplying voltages matrix; $[I]$ – current matrix; J – moment of inertia; T_e – electromagnetic torque; T_0 – load torque; f – friction coefficient; Ω – rotor angular speed; Θ – rotor angle.

Basing on above motor mathematical description, asynchronous motor state equations can be presented as follows:

$$\frac{d}{dt}[I] = -[L]^{-1}\left\{ [R] + \Omega\frac{d[L]}{d\Theta} \right\}[I] + [L]^{-1}[U],$$

$$\frac{d\Omega}{dt} = (T_e - f\,\Omega - T_0)/J, \tag{8}$$

$$\frac{d\Theta}{dt} = \Omega.$$

Form of the matrixes in those equations depends on the motor type and modeled damage states. With use of the presented library, description of squirrel-cage motor state equations in C++ language is:

```
dI=1.0/Ind.L*(U-(Res.R+Ind.dL*pulsacja)*I);
mom=(!I)*Ind.dL*I*0.5;
pulsacja^=(Me-Mo)/J;
kat_^=pulsacja;
```

where: dI – current derivatives matrix; Ind – instance of the TInductance class object – includes inductances matrix components; Ind.L and their derivatives by rotor angle kat; mom – electromagnetical torque; I – motor currents matrix; U – motor voltages matrix; Res – instance of the TResistance class object – includes resistances values matrix; pulsacja – rotor speed; kat – rotor angle; J – moment of inertia; Mo – load torque.

Above description for motor type ex. Sh90L2 contains calculations of 27 state variables derivatives. In the result of application and multiple testing of earlier described libraries, the way of those equations notation is almost similar to their mathematical notation. All the methods that are neccesary for making of the state equation calculations are placed in separate classes. Resistances calculation are placed in the *TResistance* class, and for inductances calculations the *TInductance* class is prepared. Such solution gives a possibility of independent testing of chosen components describing damaged motor without neccesity of changes in the source code describing the state equations. Modeling of the rotor cage damages can be made by changing elements of the rotor resistances matrix. Considering accuracy of inverse matrix calculation, it's recommended to increase resistance of one rotor bar or ring segment not more than 100 times. Modeling of rotor eccentricity (static and dynamic) on the other hand requires modifications of the elements of the inductance matrix [8].

Motor model, including inductances and resistances matrixes calculation classes and state equation descriptions have been placed in the *ASYNCH* class. For connection of this class components with different methods of controlling them, interface in the form of public access methods providing motor's phase voltages modifying is being used. Such solution gives a possibility of independent motor drives control algorithms testing. In shown case, control algorithms were tested on a classical motor model. Only tested inverter control algorithms are used for motor drives modeling including chosen damages.

4. Examples of numerical modeling

Described program solutions were used for many simulation programs building, aiding didactic process. Basic goal of their creation was to prepare concrete applications providing influence of many different induction motor damages on the motor run studying. Applications for dynamic analysis of motor drive run, controlled by vector methods (Field Oriented Control − FOC), non-linear methods (Direct Torque Control − DTC), scalar (U/f = const.) and sinusoidal voltage were prepared.

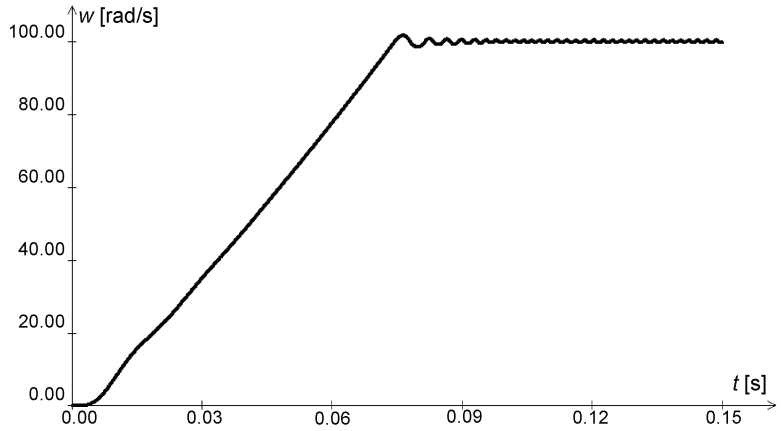

Fig. 2. Motor drive start (controlled by DTC method) to set speed value 100 rad/s

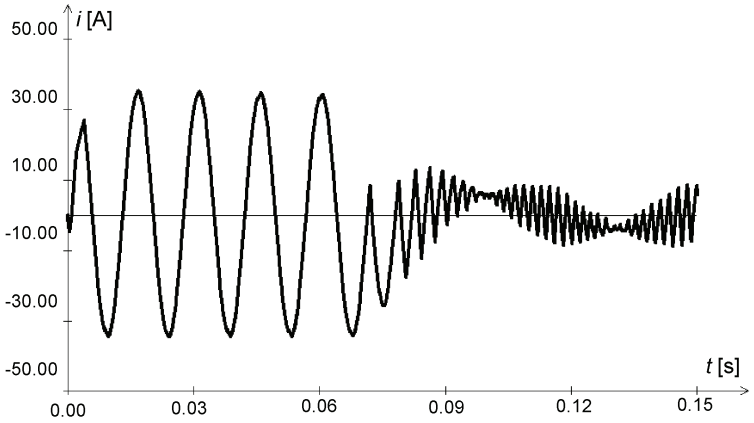

Fig. 3. Instantaneous current value during motor start – DTC method

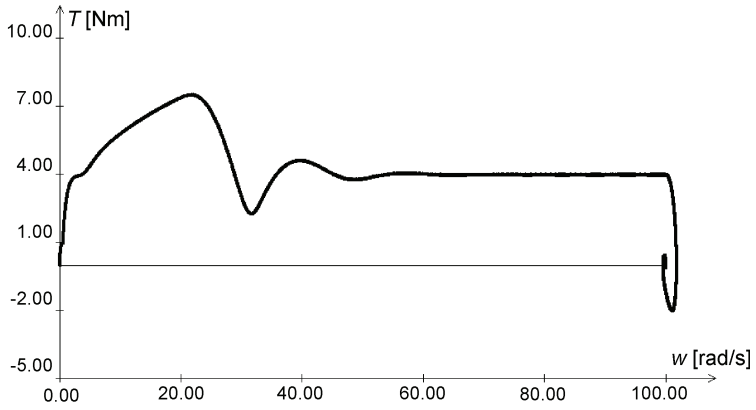

Fig. 4. Dependence of motor torque from angular speed during motor start – FOC method

In Figures 2 and 3 waveforms of squirrel-cage motor's controlled with DTC method angular speed and phase current during motor start to set speed value to 100 rad/s are shown.

Figure 4 shows dependence of motor torque from the angular speed during motor start to set speed value of 100 rad/s (FOC control method).

In Figure 5 temporary motor torque value using the field oriented control method during simulated damage of 4 consecutive rotor bars braking is shown. Figure 6 shows instantaneous estimated rotor flux value during motor start.

Designed applications provide rotor damages modeling (rotor cage and end-rings braking) and also rotor eccentricity modeling (static and dynamic).

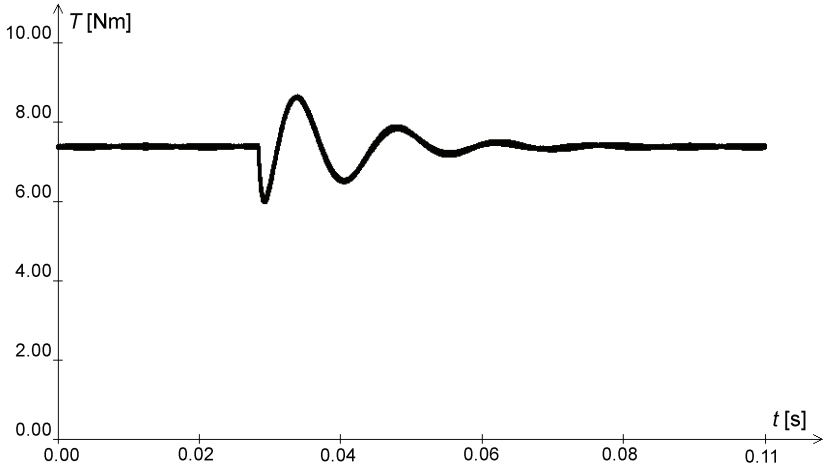

Fig. 5. Instantaneous motor torque value during the moment of 4 consecutive rotor bars braking (FOC method)

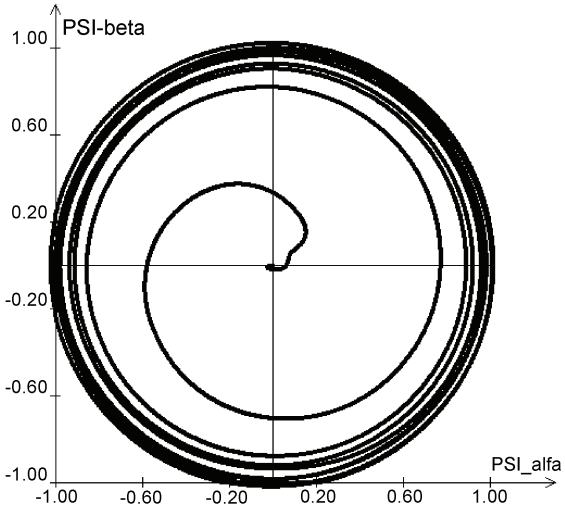

Fig. 6. Estimated rotor flux – FOC method

Actually, researches for developing applications providing other motor damages modeling (for example stator windings damages) are continued.

5. Conclusions

Using of object-oriented programming techniques greatly simplifies any physical process behaviour modeling. Due to independent user state equation programming possibility, presented class libraries provide any physical phenomenons described by differential equations modeling. Prepared class libraries provide easy squirrel-cage motors mathematical model notation, enabling easy modification of its components. Enchancing of the numerical modeling precision requires only separate model components modifications, without ingerention in the whole modeling system structure. In easy and comfortable way, the user can do any independent calculation algorithm testing – either parameter calculation algorithms, or any dynamic algorithm execution calculations. Discussed solution was used for modeling of motor drive runs with different control methods. Either programs based on a classical motor model, or the ones providing any motor damages simulating have an identical structure.

References

[1] Dattatri K., *Język C++ Efektywne programowanie obiektowe (C++: Effective Object-Oriented Software Construction.* (In Polish), Helion, Gliwice (2005).
[2] Didier G., Razik H., Abed A., *On Space Harmonic Model Of A Three Squirrel Cage Induction Motor For Diagnosis Purpose.* EPE-PEMC, Dubrownik & Cavtat (2002).
[3] Houdouin G., Barakat G., Dakyo B., Destobbeleer E., *A Method the Simulation of Inter-Turn Short Curcuits in Squirrel Cage Induction Machines.* EPE-PEMC, Dubrownik & Cavtat (2002).
[4] Joksimović G. M., *An Approach to Dynamic Simulation of Dynamic Eccentricity in Induction Machines.* EPE-PEMC, Dubrownik & Cavtat, 2002.
[5] Josuttis N.M., *C++ Biblioteka standardowa. Podręcznik programisty.* (*The C++ Standard Library: A Tutorial and References.* (In Polish), Helion, Gliwice (2003).
[6] Kluszczyński K., Miksiewicz R., *Modelowanie 3-fazowych maszyn indukcyjnych przy uwzględnieniu wyższych harmonicznych przestrzennych przepływu.* Zeszyty Naukowe Politechniki Śląskiej, Elektryka 142, Gliwice (1995).
[7] Mishra C., Routray A., Mukhopadhyay S., *Experimental Validation of Coupled Circuit Model and Simulation of Eccentric Squirrel Cage Induction Motor.* Industrial Technology ICIT: 2348-2353 (2006).
[8] Sołbut A., *Calculation of squirrel-cage motor inductances including rotor eccentricity.* Archives of Electrical Engeneering 58(3-4): 157-169 (2009).
[9] Stroustrup B., *Programowanie. Teoria i praktyka z wykorzystaniem C++.* (*Instructor's Guide for Programming Principles and Practice using C++.* (In Polish), Helion, Gliwice (2010).