Computation Code of Software Component for the Design by Optimization of Electromagnetical Devices

H. Nguyen-Xuan, L. Gerbaud, L. Gabuio, F. Wurtz
Grenoble Electrical Engineering Laboratory (G2Elab) ENSE3 (Grenoble INP-UJF, CNRS UMR 5529)
BP 46 –38402 Saint Martin d'Hères Cedex, France
Hoa.Nguyen-Xuan@g2elab.grenoble-inpg.fr

Abstract —The paper deals with the formulation and programming code architecture of the software component generated by the Reluctool software [6] for the modeling by large reluctance networks in electrical machine applications. Portable model formulation and code architectutres are introduced facing the model generation time, the model size, the computation time. Different approaches are discused. Finally, a code architecture is compared with the actual code architecture of the commercial version of Reluctool.

I. INTRODUCTION

A designer can use different kinds of models in order to size and optimize a device. Some of them, like finite-element model or boundary element method, can be very precise, but need large computation time, limiting the number of parameters and constraints that can be taken into account. So the designer should also use an approach like the reluctance network approach, when it is necessary to deal with a large number of parameters and many constraints [1]. The modeling of electrical machine by reluctance network always starts with a network building according to the paths of magnetic fluxes using a finite element analysis. The size of the network depends on the model of the magnetic circuit, the fineness level and complexity of the paths of the magnetic fluxes. In the case of electrical machines, the number of components in a reluctance network depends mainly on the number of the slots of the stator and the number of poles of the rotor. Specifically, the reluctance network of machines needs to modeling the rotation of rotor on an electrical period in order to evaluate harmonic values of the torque and the back-emf [2]. For this modeling, the airgap reluctance network should change its topology to adapt with many rotor positions [4]. At the present time, in the context of portable software component [3], the variability of airgap network needs to regenerate software component at each network topology; or the designer should predict all the possible topologies needs to calculate [4] and generate in one time the software component. These approaches induce problems to resole for Reluctool software e.g.:

- reduction of the size of the generated software component to be able to obtaining a modeling of the large reluctance networks of machine applications and integrate different topologies in a same software component,
- reduction of the time necessary to generate large models,
- acceleration of the computation model time and its derivatives,
- calculation of the necessary derivatives for the optimization algorithm behind.

In section II, sizing model generation methodology is introduced. Different approaches of formulation model and code architectures are proposed to push back the existing limitations. Finally, two different machine models are tested to compare these approaches.

II. GENERATION METHODOLOGY OF MODEL

A. Function call versus expression substitution

In order to formulate all the outputs and its derivatives, all the basic element submodels, contained in the network, are converted into equations and functions. All the equations and functions can be established, based on the direct mathematical expression substitution approach. For example, a nonlinear reluctance is calculated by $f(\varphi)=H(B).L/\varphi$ where H is the induction and calculated by Eq.1. a, J_s, μ_r are model coefficients. B is the field and calculated as function of the flux φ ($B=\varphi/S$). L and S are geometries parameters of the reluctance. So, the expression substitution gives the final expression of the nonlinear reluctance as function of flux in Eq. 2. The translated java code of Eq.2 is long.

$$H(B)=\left((\mu_r-2a+1)\cdot B-\mu_r\cdot J_s-J_s\cdot(2a-\mu_r)\cdot\sqrt{\left(\frac{(\mu_r-1)\cdot B}{J_s\cdot(2a-\mu_r)}\right)^2-\frac{4a\cdot(a-\mu_r)}{(2a-\mu_r)^2}}\right)/(2\mu_0\cdot(\mu_r-a)) \quad (1)$$

$$f(\varphi)=\left((\mu_r-2a+1)\cdot\left(\frac{\varphi}{S}\right)-\mu_r\cdot J_s-J_s\cdot(2a-\mu_r)\cdot\sqrt{\left(\frac{(\mu_r-1)\cdot\left(\frac{\varphi}{S}\right)}{J_s\cdot(2a-\mu_r)}\right)^2-\frac{4a\cdot(a-\mu_r)}{(2a-\mu_r)^2}}\right)/(2\mu_0\cdot(\mu_r-a))\cdot\frac{L}{\varphi} \quad (2)$$

Contrarily, if the function call approach is used. The reluctance formulation will be very short and compact as $f(\varphi)=H(\varphi/S).L/\varphi$. Therefore, this approach can reduce the size of all the functions and the equations, so its translated java code size. Table I comprares these approaches. The detail will be introduced in the full paper.

TABLE I
FUNTION CALL VERSUS EXPRESSION SUBSTITUTION

Functional substitutions	Functional calls
- explicit calculation - large formula	- formal and short - numerical stability

B. Scalar approach versus vectorial approach

In parallel the establishment of all the equations and functions of all the basic element submodels, the reluctance network has to be formulated. A reluctance network is represented by an implicit system $f(\psi,I)$ (Eq.3). It can be symbolically built from a set of independent meshes of the network [3].

$$f(\psi,I)=F(I)-S\cdot R(\phi,I)\cdot S^T\cdot\psi=0,\phi=S^T\cdot\psi \quad (3)$$

Here, F is the vector of the magnetomotive forces of every flux loop. S is the loop matrix. R is the diagonal matrix containing all the reluctances of the circuit. Φ is the vector of reluctance fluxes and ψ is the vector of loop fluxes. I is the vector of input parameters. In order to solve the system $f(\psi,I)$

and to calculate all the derivatives of model outputs, the generated model has to calculate all derivatives of every equation $f(\psi,I)_i$ according to its loop fluxes ($\partial f_i(\psi,I)/\partial\psi_i$, $i = 1..n$) and its inputs ($\partial f_i(\psi,I)/\partial I_j$, $i = 1..n$ and $j=1..k$) (see more detail in the [3]). All the symbolic derivatives of the model outputs according to its inputs also have to be made. Then the model is converted into a portable programming language as Java and compiled. In these works, two different approaches (vectorial or scalar) can be implemented (Fig. 1).

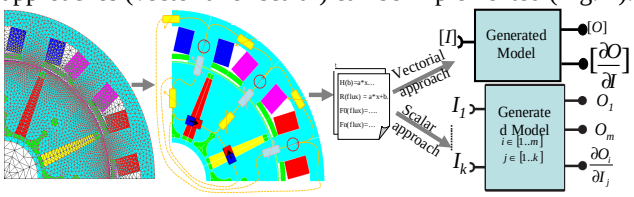

Fig. 1.Sizing model generation from reluctance network

At the present time, the scalar approach is implemented in Reluctool by other old works where all the equations of the implicit system $f(\psi,I)_i$ are found by expanding Eq.3 [5]. Then, all necessary derivatives are calculated by symbolic derivation method applied on each scalar function $f(\psi,I)_i$. Finally, each equation is translated as a java scalar method in a java package that is compiled to obtain an executable software component [5]. In fact, this scalar approach is simple to implement. However, this approach is very time consuming for the making of the derivative formulations, mostly for the large reluctance networks. In addition, the number of the generated java method increases according to the square of the implicit system size. So the size of software component generated increases significantly when the reluctance network is large. Furthermore, the compilation of the generated codes is time consuming and memory consuming due to the limitations of Java.

Contrarily, the vectorial approach can be used to formulate the model and to structure again le code architectures of the generated software component. This approach formulates $f(\psi,I)$ and its derivatives with matrix equations as Eq.3, Eq.4.

$$\begin{cases} \partial f(\psi,I)/\partial\psi = -S\cdot R(\phi,I)\cdot S^T - S\cdot(\partial R(\phi,I)/\partial\phi)\cdot\phi\cdot S^T \\ \partial f(\psi,I)/\partial I = \partial F(I)/\partial I - S\cdot R(\phi,I)\cdot S^T\cdot(\partial\psi/\partial I) - S\cdot(\partial R(\phi,I)/\partial I)\cdot\phi \end{cases} \quad (4)$$

The model generation process of any reluctance network has to calculate only the simple derivatives: $\partial R(\Phi,I)/\partial\Phi$, $\partial R(\Phi,I)/\partial I$, $\partial F(I)/\partial I$. With the vectorial formulation, $f(\psi,I)$ and its derivatives are translated into java vectorial methods. They are calculated by matrix operator. So, this vectorial approach can benefit of all the common expressions appearing in Eq.3 and Eq.4, as $S.R(\Phi,I)$. S^T and S^T, to accelerate the computation tasks. The resulting calculation sequence will be presented in the full paper. So, the computation is faster. Table II comprares these approaches. The detail will be introduced in the full paper.

TABLE II
SCALAR VERSUS VECTORIAL APPROACH

Scalar formulation	Vectorial formulation
- simplicity to implement - large size of software component - time consuming of generation model - difficulty of the rotor rotation formulation	- small size of software component - quickness of generation model - acceleration of model calculation - vectorial formulation generic - possibility of the rotor rotation formulation

III. RESULTS

In this section, the paper compares these approaches for two machine models (Claw-Pole Alternator model and 12/8 PMSM) according to the following aspects: expression substitution, scalar approach, function calls, and vectorial approach. The results are shown in Table III.

TABLE III
COMPARISON BETWEEN DIFFERENT APPROACHES

Used Method	Descriptions	Claw-Pole Alternator	12/8 PMSM
Substitution and scalar	Generated model size	2.3MB	unavailable
	Model generation time	240(s)	
	Model computation time	0.437(s)	
	Derivative computation time	0.875(s)	
Functional Call and Vectorial	Generated model size	0.5MB	1.2MB
	Compilation time	40(s)	60(s)
	Computation time	0.243(s)	0.203(s)
	Derivative computation time	0.265(s)	0.230(s)

As shown in Table III, the second approach reduces 4.6 times the generated model size and 16 times model generation time, less than the first approach with Claw-Pole Alternator model. Specifically, the first approach does not allow to generate the model of 12/8 PMSM because of code compiler memory limitation. But the second approach generates fastly this model. Furthermore, this second approach reduces also, twice the model computation time and 4 times the derivatives computation time, less than the first approach. This result will be detailed in the full paper.

IV. CONCLUSION

In the paper, different approaches formulate the derivative and to structure the code of reluctance network model have been introduced in order to decrease the generation time and the size of model generated by Reluctool. The results shown that function call and vectorial approach perform and decrease limit of generation of models and reduced also the computation time and the derivative computation time of the generated models.

ACKNOWLEDGMENTS

The authors thank the French National Reasearch Agency (ANR) for their support throughout the project 3MT.

V. REFERENCES

[1] T.Raminosoa, I.Rasoanarivo, F.-M.Sargos, R.N.Andriamalala, "Constrained Optimization of High Power Synchronous Reluctance Motor Using Non Linear Reluctance Network Modeling," *Industry Applications Conference, 41st IAS Annual Meeting. Conference Record of the IEEE*,vol.3,no.,pp.1201-1208,8-12 Oct. 2006

[2] H. Dogan, L.Garbuio, H. Nguyen-Xuan, B.Delinchant, A.Foggia and F.Wurtz, "Multistatic Reluctance Network Modeling for the Design of Permanent Magnet Synchronous Machines" *IEEE Conference on Electromagnetic Field Computation*, CEFC 2012, Oita, Japan.

[3] A. Delale, L. Albert, L. Gerbaud and F.Wurtz, "Automatic Generation of Sizing Models for the Optimization of Electromagnetic Devices, Using Reluctance Networks". *Magnetics, IEEE Transactions on* , vol.40, no.2, pp. 830- 833, March 2004

[4] Y.Tang, T. E.Motoasca; J. J. H.Paulides, E. A.Lomonova, "Analytical modeling of flux-switching machines using variable global reluctance networks," *Electrical Machines (ICEM), 2012 XXth International Conference on* , vol., no., pp.2792-2798, 2-5 Sept. 2012.

[5] B.du Peloux, L.Gerbaud, F.Wurtz, V.Leconte, F.Dorschner, "Automatic generation of sizing static models based on reluctance networks for the optimization of electromagnetic devices," *Magnetics, IEEE Transactions on* ,vol.42, no.4, pp.715-718, April 2006