From the electronic circuit to the simulation component : an automatic component building process

Vincent Fischer, Laurent Gerbaud
Laboratoire d'Electrotechnique de Grenoble, CNRS UMR 5529 INPG/UJF
ENSIEG BP 46, 38402 Saint Martin d'Hères, France
vincent.fischer@leg.ensieg.inpg.fr, laurent.gerbaud@leg.ensieg.inpg.fr

KEYWORDS

ODE solving, matrix exponentials, constraint optimisation

ABSTRACT

This paper proposes a process to obtain automatically a simulation component dedicated to optimisation purposes from the schema of an electrical, electronic or power electronics circuit. This component, which computation is based on a specific matrix exponential calculation, has the ability to give the gradients (for SQP optimisation process), and offers short computation time and low memory occupation.
After the introduction of the optimisation component problematic, the specific use of the matrix exponential for ODE solving is presented. The computation of the matrix exponential is discussed, then the complete ODE solving and the gradients computation are detailed. After, the component building process is explained. The circuit is analysed in order to obtain its equations as a state system, the calculation code is then generated, and finally the packaging of the component is done. In the last part, some results are analysed in terms of accuracy and computation time.

0. INTRODUCTION

This paper deals with the automatic building of the model of an electronic or power electronics circuit, and the integration of this model with its solving algorithm into a Java component designed in an optimisation aim. This component has to provide a computation time as short as possible, with good result accuracy, and the ability to give gradients (for SQP optimisation process). In the paper, only linear state equations are treated, as they are most of the time encountered in the power electronics area. The simulation of hybrid state systems is not taken into account, but will be easily possible in the future, as it is in fact composed of a series of linear state systems, sequenced with continuity conditions on the states.

1. THE AIM OF THE PAPER

The constrained sizing of an electrical device can be achieved through an optimisation process, which is based on a sizing model. Such a process is characterised as shown in Fig. 1 :

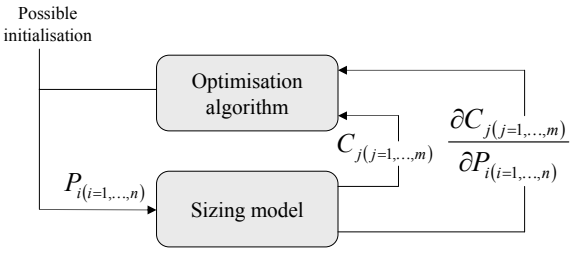

Figure 1: the optimisation process

The outputs C_j of this model are the values of the sizing criteria. They may concern values of state variables (mainly current in inductors and voltages in capacitors) at a specific date, extrema, r.m.s. or average values, etc… These criteria are calculated from the inputs P_i of the sizing model, mainly the parameters of the device. These parameters are inductances, capacitances, resistances, etc… The sizing criteria may depend on the state variables X of the application to size. In the context of the sizing of a power electronic structure, a linearity hypothesis is often possible for the state equation which is defined by (1).

$$\dot{X}(t) = A \cdot X(t) + B \cdot u(t) \qquad (1)$$

Here, t represents the time, and $u(t)$ is the expression of the state inputs (e.g.: the sources of the electronic circuit). A, B, and $u(t)$ depend directly on the P_i, the parameters of the circuit.

Furthermore, some optimisation algorithms are based on gradient methods (e.g. VF13 [7]), which need the partial derivatives of the sizing criteria according to the model inputs, i.e. $\frac{\partial C_j}{\partial P_i}$. These derivatives can be expressed with the partial derivatives of the state variables, $\frac{\partial X(t)}{\partial P_i}$.

Different approaches exist to obtain the states values from the inputs P_i [1][2]. They are often

Proceedings 15th European Simulation Symposium
Alexander Verbraeck, Vlatka Hlupic (Eds.)

ISBN 3-936150-28-1 (book) / 3-936150-29-X (CD)

based on numerical simulation using ODE solving algorithms like Trapeze, Runge-Kutta, etc. Obtaining the partial derivatives of the states can be achieved through finite differences. These numerical approaches lead to important computation times, especially if the values of state variables have to be estimated only at a specific date. In the same way, if the sizing model has many outputs depending on several inputs, the computation of the gradient of the state variables according to the model inputs may be very time consuming and numerically sensitive.
Another approach can be considered. As electronic circuit state systems are often linear ones, the ODE solving based on matrix exponentials is a good way to reduce the computation times, as shown below.

The aim of the paper is to propose a tool that generates dedicated calculation components of linear state equations, specifically in the case of electrical circuits. Such a component allows calculating the state variables at a given date, without having to run a complete simulation. It also calculates the partial derivatives of the state variables according to the physical parameters of the state equations (fig. 2), without the need of a second evaluation, as the finite differences method needs.

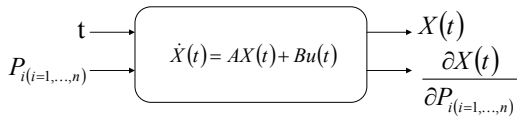

Figure 2: the calculation component

Computation time and memory occupation are as low as possible during the use of this component.

2. THE MATRIX EXPONENTIAL FOR ODE SOLVING

As in the paper, the state equation is supposed to be linear, its complete solution can be expressed by the following expression [3] :

$$X(t) = e^{At} \cdot X(0) + \int_0^t e^{A(t-\tau)} \cdot B \cdot u(\tau) \cdot d\tau \quad (2)$$

In order to value such an expression, the exponential of the matrix A has to be defined. Several algorithms can be used to estimate it [4] [5]. Some of these use the eigenvalues of A, but they are limited for our purposes. In this paper, the selected algorithms do not use such methods. They are fast and require few memory space.
Two methods have been selected : the Taylor series development and the Padé approximation. Both may be used in our tool. There are slight differences between the results obtained with each method, but these differences are negligible. The use cases of each method are determined by the state system expression. For very large state systems, the Padé approximation will be privileged, when for smaller systems, the Taylor series will be used.

2.1. The Taylor Series Development

The first selected algorithm is based on the Taylor Series Development of the exponential operator :

$$e^A = \sum_{n=0}^{+\infty} \frac{A^n}{n!} \quad (3)$$

Computing this infinite sum is obviously impossible. An upper bound N must be chosen, depending on the desired calculation accuracy ε.
To obtain the accuracy ε, the upper bound N must satisfy to the following criteria [4]:

$$0 < \left(\frac{\|A\|^{N+1}}{(N+1)!} \right) \left(\frac{1}{1 - \frac{\|A\|}{(N+2)}} \right) \leq \varepsilon \quad (4)$$

As long as the Frobenius norm ($\sqrt{Trace(A^T \cdot A)}$) of the matrix A is smaller than 1, this algorithm gives accurate results, but when the norm is greater than 1, the accuracy is lost.
To compensate for that issue, the scaling and squaring method is used. This method is based on the following expression :

$$e^A = \left(e^{\frac{A}{2^S}} \right)^{2^S} \quad (5)$$

S is chosen so that $\left\| \frac{A}{2^S} \right\| < 1$. Then the exponential of $\frac{A}{2^S}$ is computed, and finally the result is squared S times. This algorithm gives very accurate results for all kinds of matrixes, as long as they are well conditioned.
The condition number of a matrix measures the sensitivity of the solution of a system of linear equations to errors in the data. It also indicates the numerical disparity between the elements of the matrix. The condition number is calculated as shown in equation (6) :

$$Cond(A) = \frac{V_{S_{MAX}}}{V_{S_{MIN}}} \quad (6)$$

where $V_{S_{MAX}}$ is the highest singular value of the matrix, and $V_{S_{MIN}}$ the smallest one.
When the condition number of a state system matrix is too high, it generally comes from the fact that two different modelling levels have been mixed. For example, mixing the first order modelling with the HF EMC modelling gives components with very disparate numerical values (the first order model will lead to components with big numerical values, as the HF EMC model will lead to very small numerical values). The models of

circuit must be homogeneous on the modelling level. For example, to study the global functioning of a system, the first order model will be used, while the EMC will be studied with a specific HF EMC model.

2.2. The Padé Approximation

The (p,q) Padé approximation is based on the following expressions (7):

$$N_{p,q}(A)=\sum_{i=0}^{p}\frac{(p+q-i)!\,p!}{(p+q)!\,i!\,(p-i)!}A^i$$
$$D_{p,q}(A)=\sum_{i=0}^{q}\frac{(p+q-i)q!}{(p+q)!\,i!\,(q-i)!}(-A)^i \qquad (7)$$
$$R_{p,q}(A)=\left[D_{p,q}A\right]^{-1}\left[N_{p,q}(A)\right]$$

It gives accurate results with reduced computation time [6] as long as the norm of the matrix is small. So, as for the Taylor series development, the scaling and squaring technique has to be used. The results are quite equivalent with the Taylor series methods. There are slight differences in terms of computation time or memory occupation, but it depends on the matrix of which the exponential is computed, and the differences are negligible.

2.3. Obtaining the complete solution of the state equation

As the considered systems are power electronics ones, the electrical sources are either constant or sinusoidal.
As it will be detailed in the following part, equation (1), may be derivated according to the input parameters (e.g.: inductance, resistance, etc), giving a new state equation where expression of u(t) may be constant, sinusoidal, or the product of a polynomial expression with a sinusoidal expression.
If $u(t)=U$ (e.g. a constant source), then :

$$\int_0^t e^{A(t-\tau)}\cdot B\cdot u(\tau)d\tau=\left[A^{-1}\cdot e^{At}-A^{-1}\right]B\cdot U \qquad (8)$$

This gives:

$$X(t)=e^{At}\cdot X(0)+A^{-1}\left[e^{At}-I\right]B\cdot U \qquad (9)$$

For a sinusoidal source (e.g. $u(t)=sin(\omega t)$), the integral term of the solution becomes :

$$\left[A^2-\omega^2 I\right]^{-1}\begin{bmatrix}e^{At}\left[\omega cos(\varphi)+sin(\varphi)\right]-\\ \left[\omega cos(\omega t+\varphi)+Asin(\omega t+\varphi)\right]\end{bmatrix}B \qquad (10)$$

For a polynomial source, (e.g. $u(t)=\sum_{k=0}^{d_U}c_k t^k$ where d_u is the degree), the integral term is:

$$\sum_{k=0}^{d_u}\left[-c_k A^{-(k-1)}\sum_{n=0}^{k}\left[k!\,e^{At}+\frac{(At)^n k!}{n!}\right]\right]B \qquad (11)$$

Finally, all the different solutions for every single source have to be added to obtain the complete solution of the state system:

$$X(t)=e^{At}\cdot X(0)+\sum_{i=1}^{N_S}B_i\int_0^t e^{A(t-\tau)}u_i(\tau)d\tau \qquad (12)$$

where N_S is the number of sources.

2.4. Obtaining the states partial derivatives

Obtaining the states partial derivatives can be achieved by numerical methods such as finite differences, which shall be avoided as these methods use more computation time and memory. Moreover, these methods are numerically very sensitive to their adjustment parameters.
Another method giving the states partial derivatives is based on a system symbolic recombination. In this way, the system equation is:

$$\dot{X}(t)=A\cdot X(t)+B\cdot u(t) \qquad (13)$$

The derivative of this equation according to any input P_i of the sizing model gives:

$$\frac{\partial\dot{X}(t)}{\partial P_i}=\frac{\partial A}{\partial P_i}\cdot X(t)+A\cdot\frac{\partial X(t)}{\partial P_i}+\frac{\partial B}{\partial P_i}\cdot u(t)+B\cdot\frac{\partial u(t)}{\partial P_i} \qquad (14)$$

A new system can be created by these two equations:

$$\dot{\widetilde{X}}=\widetilde{A}\cdot\widetilde{X}+\widetilde{B}\cdot\widetilde{u}(t) \qquad (15)$$

where :

$$\widetilde{X}(t)=\begin{bmatrix}X(t)\\ \dfrac{\partial X(t)}{\partial P_i}\end{bmatrix},\ \widetilde{A}=\begin{bmatrix}A & 0\\ \dfrac{\partial A}{\partial P_i} & A\end{bmatrix},\ \widetilde{B}=\begin{bmatrix}B & 0\\ \dfrac{\partial B}{\partial P_i} & B\end{bmatrix}$$

and $\widetilde{u}(t)=\begin{bmatrix}u(t)\\ \dfrac{\partial u(t)}{\partial P_i}\end{bmatrix}$.

This new equation system is linear, and can be solved with the methods presented in this paper. In order to obtain the gradients of the states, the state equation has to be derivated according to every input of the sizing model, e.g. the P_i where $i=\{1,\ldots,N_D\}$.To obtain the lowest computing time, a new system will be created and solved for each derivate calculated. A system containing all the derivates could be created with the following matrixes:

$$\widetilde{X}(t)=\begin{bmatrix}X(t)\\ \dfrac{\partial X(t)}{\partial P_1}\\ \vdots\\ \dfrac{\partial\dot{X}(t)}{\partial P_{N_D}}\end{bmatrix},\ \widetilde{A}=\begin{bmatrix}A & & & \\ \dfrac{\partial A}{\partial P_1} & A & & (0)\\ \vdots & & \ddots & \\ \dfrac{\partial A}{\partial P_{N_D}} & (0) & & A\end{bmatrix},$$

$$\widetilde{B}=\begin{bmatrix} B & & \\ \dfrac{\partial B}{\partial P_1} & B & (0) \\ \vdots & & \ddots \\ \dfrac{\partial B}{\partial P_{N_D}} & (0) & B \end{bmatrix} \text{ and } \widetilde{u}(t)=\begin{bmatrix} u(t) \\ \dfrac{\partial u(t)}{\partial P_1} \\ \vdots \\ \dfrac{\partial u(t)}{\partial P_{N_D}} \end{bmatrix}.$$

With this system, only one solving is needed to obtain all the derivatives of the states. But the solving of this system needs the exponential of the matrix $\widetilde{A}$, of which the computation time proportional to $\left((N_D+1)*n\right)^3$ where n is the size of A and N_D the number of derivatives computed. When the derivatives are calculated separately from each other, the total computation time is proportional to $(n*2)^3*N_D$. Solving several small systems requires less computation time than solving one large system.

3. THE COMPONENT BUILDING

The process leading from the circuit schema to the simulation component is divided in three distinct parts, as shown in Fig. 3. First of all, the current equations are obtained from the circuit scheme, and the state system is extracted from the circuit equations. The computation code is then generated. Finally, the component is packaged.

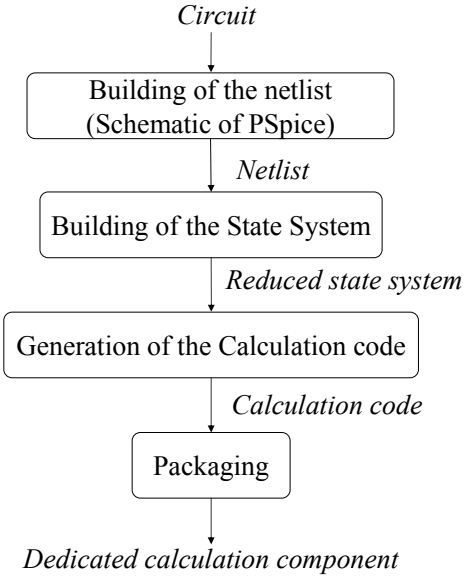

Figure 3: the component building process

3.1. Expressing the state system

The circuit is described using Pspice Schematic. From this tool, a netlist describing the circuit is obtained and can be used by a builder which gives the reduced state equation of the circuit.

This builder parses the PSpice netlist to extract the node equations of the circuit. Form that, an independent set of mesh equations is built and the equations of each components are given [8]. The modelling level used in a sizing process allows having simple linear models for the components (resistors, inductors, capacitors).

Having all the equations of the circuit, the reduced state equations are obtained using symbolic treatment implemented in Macsyma [9].

The time differentiated variables of a circuit are the currents in its inductors and the voltages on its capacitors. Thanks to the algebraic relations between the voltages of the circuit (from the mesh equations) and the currents in the circuit (from the node equations), the state variable vector is only made of a part of these variables.

In this way, the state equation is defined by equation (1). The state variable vector X is made with currents in inductors and the voltages on capacitors. The input vector $u(t)$ is made with the circuit sources. Finally, the coefficients of the state variables and the sources in each equation are symbolically computed, defining the matrixes A and B. So, the state system is symbolically expressed.

3.2. Automatic building of the calculation code

With the expression of the state system, the model calculation code can be generated automatically. In this process, such a component calculates the state variables at a given date, without having to run a complete simulation. It can also calculate the partial derivatives of the state variables according to the physical parameters of the circuit (e.g. resistances, inductances, capacitances, etc.) (see fig. 4).

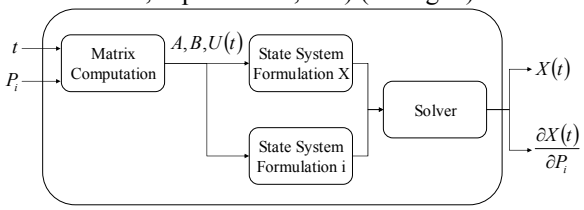

Figure 4: Structure of a generated calculation component

The created calculation component is Java based, but it uses some algorithms written in C (the matrix exponential algorithm for example). The generated Java code calls the C code during the calculation of the model. First of all, the calculation code of the state system matrixes (A, B, and $u(t)$) can be generated without any particular difficulty, as their expressions are ANSI-C compliant. The generation of the calculation code of the partial derivatives of the state system matrixes requires a symbolic derivation treatment. In the paper, a Java based lightweight derivation tool developed in our laboratory (RAMA [11]), has been used.

Then the code using the calculation of the matrix exponential and the calculation of the state system

matrixes is generated. This code can calculate the value of the state vector X at any date, without simulating any transient state. Finally, the specific code of the calculation component is generated. This last code includes the definition of the inputs and outputs of the component. It also includes the default implementation of the simulation component interface. This interface contains the methods necessary to access to its inputs and outputs (reading and writing their values, and connecting an input to an output of another component). It also contains two computation methods. The first one launches the computation of the outputs with the actual values of the inputs. The second one launches the computation of the partial derivatives of the outputs according to the specified inputs. All of these methods are implemented automatically. The code is then compiled.

3.3. Packaging the component

All the generated Java classes and the dynamic library containing the C functions (matrix exponential computation, matrix inversion, matrix determinant, …) used by the Java code are included in a Jar (Java ARchive) file containing a manifest pointing to the main class of the component, which is the class implementing the component interface. During the loading of this class, all the other classes (which are used by the component main class) are loaded, as well as the dynamic libraries, which contain the C code.
The component is ready to compute the model, or to be connected with others components…

4. RESULTS

The results obtained with a static converter plugged on an electronic circuit representing the RSIL filter of a power electronics converter (fig. 5) is presented to illustrate the approach. The results given by the generated calculation component are compared with numerical simulations, in terms of accuracy, computation time and memory occupation. Simulations have been made with Simplorer [10].

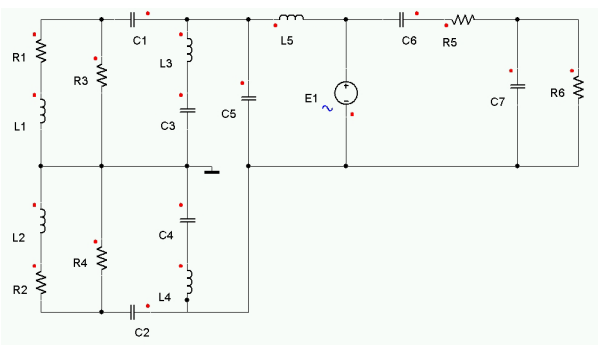

Figure 5: the electronic circuit

The generation of the component is done in less than one minute, including the analysis of the circuit, the generation of the state equation, the generation of the computation code and the packaging into a component.
The comparison is done for several frequencies of the voltage source, to get a good representation of the harmonic spectrum of the operating mode.
Although the whole simulation takes a little more computation time with the matrix exponential approach than with a numerical approach, the time gain lies in the fact that each point of the simulation can be obtained without the knowledge of the others, whereas the numerical simulation needs the knowledge of the past to obtain the next point. It means that obtaining the states values at $t = 1s$ will require at least thousands of calculated points with numerical integration methods, whereas only one resolution is needed with the matrix exponential approach. As a consequence, the computation is far faster and the memory occupation far lower with the matrix exponential method than with the numerical simulation. The accuracy is equivalent with both methods, as shown in Fig. 6.

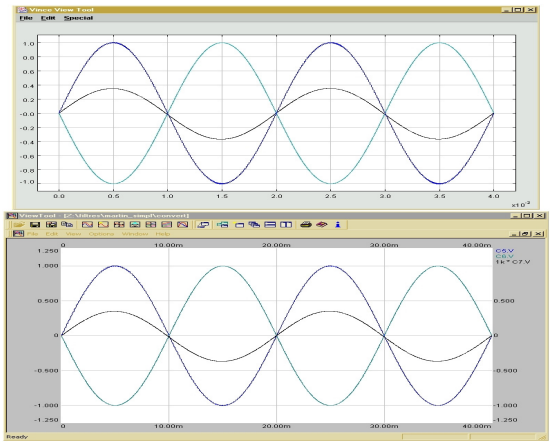
Figure 6: the results of the matrix exponential approach (up), and the numerical simulation (down)

Moreover, the calculation component gives also the gradients for optimisation processes.

5. CONCLUSION

In this paper we have presented a process which builds automatically a Java based component containing a model of an electronic circuit, and the algorithms required for the computation of this model. The component is automatically generated from the schema of the circuit.
The computation of the model is based on a methodology using symbolic treatments of the models and matrix exponential to solve the ordinary differential equations involved in the models of electronic circuits. This method gives directly the values of the states at a specific date and their partial derivatives according to the circuit parameters, for optimisation purposes. To the contrary, numerical methods must start from the

origin and integrate the ODE step by step until they reach the desired date. To obtain the partial derivatives of the states with the finite differences, another simulation must be computed for each input of the model.

The presented methodology requires few computing time and few memory occupation. The states are obtained with a very good accuracy, as well as the partial derivatives, to the contrary of the derivatives obtained with finite differences, which are numerically sensitive to the differentiation step, and which may be unstable.

The purpose of this methodology based on matrix exponentials is not the simulation of electrical devices, as other tools can do it very well, but to reduce the computing time while obtaining the values of the states at a certain date and their partial derivatives. This methodology shall reduce the overall computing time of the optimisation of an electrical device.

6. REFERENCES

[1] Kragh H., Blaabjerg F., Pedersen J.K., "An advanced tool for optimised design of power electronic circuits", proceeding of IEEE-IAS'98, Saint Louis, Missouri, USA, October 12-15, 1998, pp 991 – 998

[2] Viarouge P., Tourkhani F., Kamwa I., Le-Huy H., "Nonlinear optimization techniques for the design of static converters", proceedings of the IMACS-TC1'93, 4th international conference, Association for Mathematics and Computers in Simulation, Montreal, Canada, July 7-9, 1993, pp 543 – 547

[3] J. D'Azzo, C. Houpis, "Linear Control System Analysis and Design", 4th Ed., McGraw Hill Book Co., 1995

[4] C. Moler, C. Van Loan, "Nineteen dubious ways to compute the exponential of a matrix", SIAM Review 1978, Vol. 20 No.4, pp 801 – 836

[5] W. Harris Jr, J. Fillmore, D. Smith, "Matrix Exponentials – Another Approach", SIAM Review 2001, Vol. 43 No 4, pp 694 – 706

[6] V. Fischer, L. Gerbaud, J. Bigeon, "Solving ODE for optimisation: Specific use of the matrix exponential approach", OIPE 2002, Lodtz, Poland

[7] VF13, http://www.cse.clrc.ac.uk/nag/hsl/

[8] C. Lechevalier, L. Gerbaud, J. Bigeon, "Automatic design of discrete time-models of static converter", Simulation in Industry, 8th SCS-ESS'96 (Europeen Simulation Symposium), Genoa, Italy, October 24-26, 1996, pp 475-479.

[9] L. Gerbaud, J. Bigeon, G. Champenois, "Modular approach to describe electromechanical systems. Using Macsyma to generate global approach simulation software", Conference record of the IEEE PESC'92 (Power Electronics Society Conference), june 29 - july 3, 1992, Toledo, Spain, pp 1189-1196

[10] Simplorer, http://www.ansoft.com/products/em/simplorer/

[11] V. Fischer, L. Allain, "RAMA : a lightweight rule-based tool for expressions analysis and code generation", ESS 2003, 26 – 29 October 2003, Delft, The Netherlands