THE SIMULATION OF PARTIAL DIFFERENTIAL EQUATIONS AND SYSTEM MATRICES

E.B. RUDNYI & J.G. KORVINK

Simulation Chair, IMTEK, Freiburg University, Germany
E-mail: rudnyi@imtek.de
E-mail: korvink@imtek.de

I.1.1.1 Introduction

Domain simulators for process, equipment, device and environment simulations play essential role in MEMS engineering. Mathematically speaking, they solve coupled, multi-field partial differential equations (PDEs). In most cases, the time-dependent solution of these equations is performed by a semidiscretization method, which converts PDEs to ordinary differential equations (ODEs) in time expressed in terms of so-called system matrices.

In the present work, the importance of having standards for the system matrices is emphasized. One of the reasons is that there are modern mathematical methods for the approximation of large-scale dynamic systems (automatic model reduction) and in order to employ them one needs an easy access to the system matrices.

The main problem is that in the general case the system matrices contain not numbers but rather arbitrary functions. This poses a challenge to developing a standard that can include any possible case and at the same time allows developers to implement it efficiently. The ways to solve this problem are discussed.

I.1.1.2 System Matrices and Integration in Time

In general case, partial differential equations are solved numerically by performing their discretization based on a finite difference, finite volume, finite or boundary element scheme [1]. A conventional approach for a time-depended problem is first to make a semidiscretization in space. This converts PDEs to

ordinary differential equations of a high dimension in time, which can be integrated during the second step.

As a result, the whole simulation can be split to the two relatively independent stages. During the first stage, a user builds a solid model of the device in question, meshes and discretizes it in space according one of the methods listed above. The final result is a system of ODEs of the second order that can be written down as follows:

$$M\ddot{x} + K\dot{x} + Cx = F\ , \qquad (1)$$

where M, K, and C are the system matrices, F is the load vector and x is the state vector with unknown functions in time. In principle, the dimension of the state vector may not be constant (mesh adaptivity in time, the birth and death of the elements, and so on) but we will not consider such a case in the present work.

The fact that the simulation is naturally divided into the two stages is used in order to break the problem to individual parts and to solve them individually. Thereafter, it is important to have a computer standard to express Eq (1), that is, the system matrices.

The mathematical basis for the discretization in space and the integration in time is quite different. As a result, the algorithms and their software implementation are also different and their development requires quite a different expertise. If the computer standard for the system matrices were available, this would allow different developers to concentrate on a particular problem and thus to speed up the development cycle.

At present, the bottleneck to simulate PDEs is the second stage that takes the most computational time because of the high dimensionality of Eq (1). The size of the state vector can routinely reach hundreds thousands, especially in the case of 3D-simulations, and the integration in time can take hours of processor time even on the modern computers. This prevents a system type simulation of the whole chip based on the physics laws.

The approximation of large-scale dynamic systems is a rapidly developing area [2][3]. It can be used to reduce the dimensionality of Eq (1) within acceptable error bounds and thus to obtain compact models which can be solved quickly at the system simulation level and at the same time without sacrificing the precision. An alternative approach can be tied with fast integrators based on the approximation of the matrix exponential [4]. In any case, the development of these tools requires an easy computer access to the system matrices in Eq (1) for MST applications.

I.1.1.3 Stroring System Matrices

With an exception of the boundary element method, the system matrices are sparse and it is necessary to take this into account. However now this is not a major problem as this was an active area of research. There are well-known techniques to store a sparse constant matrix [5]. This can be used for linear dynamic systems, when the system matrices are composed just from numbers. In the case of nonlinear systems, the sparsity can be preserved within so-called an element-by-element approach [6] because in this case the assembling the global matrices may require symbolic manipulations.

The principal problem is that in the general case the components of the system matrices are some functions and this poses the main challenge: how to combine universality and efficiency. Let us briefly consider possible solutions to this problem. We will follow Ref. [7] in which this was discussed in relation with computational thermodynamics.

At present, the interpretation of an ASCII expression at run-time is a popular solution to the problem stated above. There are quite developed general mathematical languages like Mathematica or Matlab, which allow us to write in the computer format any mathematical statement. The interpretation is our current solution for our research project on automatic model reduction of nonlinear dynamic systems [8]. The drawback is loss of efficiency. Pre-compilation or “just-in-time-compiling” can improve run-time effectiveness but nevertheless may require a considerable amount of time during parsing.

It is possible to make a very efficient implementation including parsing if one fixes the functional form and allows a user just to change some coefficients within it. Unfortunately, it is highly unlikely that a single functional form will be enough for all MST cases.

Thereupon, a natural extension is to allow for several functional forms. This means that there are several precompiled and optimized functions for different functional forms and the choice between them is made at the run-time according to some field within the data structure. Of course, this should be open to add new functional forms implemented in new functions at any time. Object-oriented programming permits us to have an efficient software implementation based on a virtual function call that is easy to maintain. Under object-oriented programming, the field to distinguish between different functions is even inserted automatically (a pointer to a virtual function table).

The overhead of a virtual function call is pretty small. It can be made completely negligible in our case if we group the system matrix components according to their functional forms and make a single virtual function call for the whole group.

In our view, XML presents a good framework to develop an external format to support the several functional forms. Examples in XML in this respect are presented elsewhere [7].

I.1.1.4 Conclusion

It is shown that the standard for the system matrices will stimulate research in the area of automatic model order reduction and fast integrators for Eq (1). A practical step toward such a standard should include a choice of the most important functional forms for components of the system matrices encountering during simulation of MST devices. XML is a good choice for an external representation that is easy to parse and open to new extensions.

I.1.1.5 Acknowledgement

This work is partially funded by the EU through the project MICROPYROS (IST-1999-29047), partially by the DFG project MST-Compact (KO-1883/6) and partially by an operating grant of the University of Freiburg.

I.1.1.6 References

[1] J. G. Korvink, E. B. Rudnyi, A. Greiner, Z. Liu, MEMS and NEMS Simulation, in MEMS Handbook, 2003, to be published.
[2] A. C. Antoulas, D. C. Sorensen, Approximation of large-scale dynamical systems: An overview, Technical Report, Rice University, Houston, 2001, http://www-ece.rice.edu/~aca/mtns00.pdf.
[3] E. B. Rudnyi, J. G. Korvink, Review: Automatic Model Reduction for Transient Simulation of MEMS-based Devices, Sensors Update, 2002, v. 11.
[4] M. Hochbruck, C. Lubich, H. Selhofer, Exponential integrators for large systems of differential equations, SIAM J. Sci. Comput. 1998, 19, 1552-1574.
[5] Matrix market, http://math.nist.gov/MatrixMarket/.
[6] T. J. R. Hughes, I. Levit, J. Winget, An element-by-element solution algorithm for problems of structural and solid mechanics, Computer Methods in Applied Mechanics and Engineering 1983, 36, 241-254.

[7] E. B. Rudnyi, Computational Thermodynamics Library: TDLIB'00, http://www.chem.msu.su/~rudnyi/tdlib/.
[8] J. Lienemann, B. Salimbahrami, B. Lohmann, Draft for the Dynamic System Interchange Format, http://www.imtek.uni-freiburg.de/simulation/mstkmpkt/models/DSIF-draft.html,