Model Order Reduction for Large Scale Engineering Models Developed in ANSYS

Evgenii B. Rudnyi and Jan G. Korvink

IMTEK, Institute of Microsystem Technology
Freiburg University
Georges-Köhler-Allee, 103
D-79110, Freiburg, Germany
{rudnyi,korvink}@imtek.de
http://www.imtek.uni-freiburg.de/simulation/

Abstract. We present the software `mor4ansys` that allows engineers to employ modern model reduction techniques to finite element models developed in ANSYS. We focus on how one extracts the required information from ANSYS and performs model reduction in a C++ implementation that is not dependent on a particular sparse solver. We discuss the computational cost with examples related to structural mechanics and thermal finite element models.

1 Introduction

The model order reduction of linear large-scale dynamic systems is already quite an established area [1]. In many papers (see references in [2]), advantages of model reduction have been demonstrated for a variety of scientific and engineering applications. In the present work, we focus on how engineers can combine this technique with existing commercial finite element software in order to

- Speed up a transient or harmonic analysis,
- Generate automatically compact models for system-level simulation,
- Incorporate finite element packages during the design phase.

Model reduction is conventionally applied to a large-scale dynamic system of the first order as follows

$$\begin{aligned} E\dot{\boldsymbol{x}} &= A\boldsymbol{x} + B\boldsymbol{u} \\ \boldsymbol{y} &= C\boldsymbol{x} \end{aligned} \tag{1.1}$$

where A and E are system matrices, B is the input matrix, C is the output matrix. The aim of model reduction is to generate a low-dimensional approximation to (1.1) in a similar form

$$\begin{aligned} E_r\dot{\boldsymbol{z}} &= A_r\boldsymbol{z} + B_r\boldsymbol{u} \\ \boldsymbol{y} &= C_r\boldsymbol{z} \end{aligned} \tag{1.2}$$

that describes well the dependence of the output vector $\boldsymbol{y}$ on the input vector $\boldsymbol{u}$ and so that, at the same time, the dimension of the reduced state vector $\boldsymbol{z}$ is much less than the dimension of the original state vector $\boldsymbol{x}$.

J. Dongarra, K. Madsen, and J. Waśniewski (Eds.): PARA 2004, LNCS 3732, pp. 349–356, 2006.

After discretization in space of the partial differential equations describing a user model, a finite element package generally produces a system of ordinary differential equations. At this stage, it is possible to directly apply modern model reduction methods [1]. However, the extraction of the system matrices from a commercial package happens not to be straightforward and here we share our experience on how it can be done with ANSYS [3].

We have chosen the Matrix Market format [4] to represent the reduced model (1.2). We suppose that its simulation will be done in another package, such as Matlab or Mathematica. Functions to work with the reduced model in Mathematica are available at the IMTEK Mathematica Supplement at `http://www.imtek.uni-freiburg.de/simulation/mathematica/IMSweb/`.

The system matrices are high-dimensional and sparse. As a result, the implementation of a model reduction algorithm usually depends on a particular sparse solver and a storage scheme for sparse matrices. We discuss a C++ interface that allows us to isolate the model reduction and sparse solvers completely for negligible overhead.

Finally, we analyse the computation cost and give the performance results for a few ANSYS models. The comparison of the accuracy of reduced models in respect to the original ANSYS models is given elsewhere [5].

2 mor4ansys

The developed software [6] comprises two almost independent modules (see Fig. 1). The first reads a binary ANSYS file and assembles a dynamic system in the form of Eq (1.1) for first order systems or

$$\begin{aligned} M\ddot{\boldsymbol{x}} + E\dot{\boldsymbol{x}} + K\boldsymbol{x} &= B\boldsymbol{u} \\ \boldsymbol{y} &= C\boldsymbol{x} \end{aligned} \tag{2.3}$$

for second order systems, where M, E and K are the three system matrices. The second module applies the model reduction algorithm to Eq (1.1) or (2.3), that is, it finds a low-dimensional basis V so that the approximation

$$\boldsymbol{x} = V\boldsymbol{z} + \boldsymbol{\epsilon} \tag{2.4}$$

allows us to reproduce the transient behaviour of the original state vector within the error margin $\boldsymbol{\epsilon}$.

After that, the original equations are projected to the subspace found, for example for Eq (1.2) we have $E_r = V^T EV$, $A_r = V^T AV$, $B_r = V^T B$, $C_r = CV$.

We support three methods to treat second-order systems. When the damping matrix is modeled as Rayleigh damping $E = \alpha M + \beta K$, the method from Ref [7] allows us to preserve the coefficients α and β as parameters in the reduced model. In the general case, one can choose between the transformation to a first-order system, and second order Arnoldi algorithm (SOAR) [8].

The software can also read, as well as write, the matrices for the original system in the Matrix Market format [4]. A number of model reduction benchmarks has been obtained from ANSYS by means of `mor4ansys` [9].

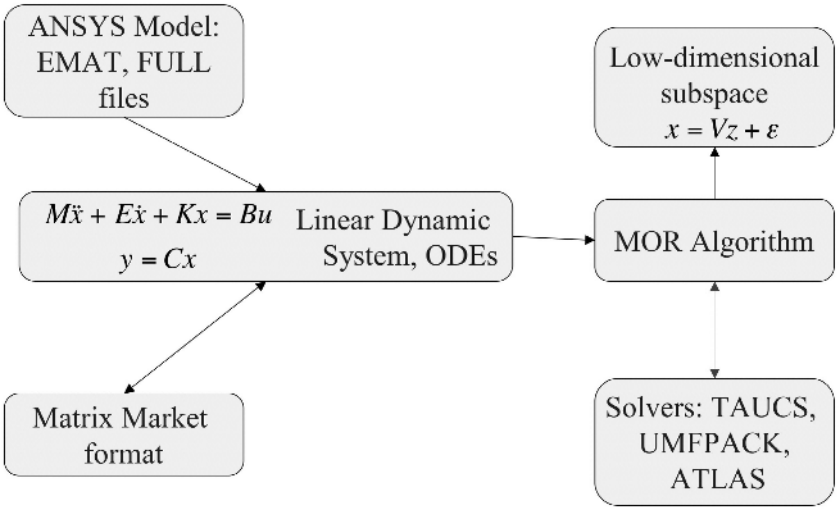

Fig. 1. `mor4ansys` block-scheme

2.1 Interfacing with ANSYS

The development of the first module happen to be rather difficult because most users of a commercial finite element package do not need the capability to extract the dynamics system in the form of Eq (1.1) or (2.3) and, as a result, this is not a trivial operation.

ANSYS is a huge package and its behavior is not completely consistent. For example, the information described below is not applicable for the fluid dynamics module FLOTRAN.

Our software reads the binary `EMAT` file with element matrices in order to assemble global system matrices. The file format is documented and ANSYS supplies a library of Fortran subroutines to work with it [10]. An example of how one can use them can be found in the `mor4ansys` code [6]. ANSYS has a special command, called a partial solve `PSOLVE`, with which one can evaluate element matrices for a given state vector without going through the real solution stage. This allows us to generate an `EMAT` file efficiently for a given model. However, it was necessary to overcome the following problems:

- The `EMAT` file does not contain the information about either Dirichlet boundary conditions or equation constraints. They should be extracted separately.
- The `EMAT` file has a contribution to the load vector from element matrices only. If nodal forces or accelerations are used to apply the load, this information should also be extracted individually.
- It is necessary to assemble the global matrices from the element matrices.

During the solution phase, ANSYS can write a binary `FULL` file with the assembled system matrices. When we started the development with ANSYS 5.7, this file did not contain the load vector (input matrix). Since then there have been many changes. Since

ANSYS 6.0 the FULL file maintains all the original matrices, the load vector, the Dirichlet and equation constraints in the file. ANSYS 8.0 allows us to make the assembly only and write the FULL file without a real solution phase (equivalent to a partial solution with EMAT). One can now also dump the information from the FULL file in the Harwell-Boeing matrix format. Hence, since ANSYS 8.0, it is possible to use the FULL file efficiently. However, depending on the analysis type the FULL file may contain not the original stiffness matrix, but rather, a linear combination of system matrices instead.

In the current version of mor4ansys, the EMAT file is employed as the main source to build Eq (1.1) or (2.3). Additional information on the Dirichlet and equation constraints and nodal forces is written in the form of text files by means of ANSYS macros we have developed. The FULL file can be used to extract the load vector when otherwise this is difficult, for example, as in the case when the acceleration load is used.

ANSYS cannot write several load vectors into the FULL and EMAT files. When multiple-input is to be preserved in Eq (1.1) or (2.3), a user should for each input:

- Delete the previously applied load,
- Apply a new load,
- Generate matrices.

In order to ease this process, the second strategy is also allowed when a user does not have to delete the previous load. In this case, each new load vector contains all the previous vectors and mor4ansys corrects them at the end of the first phase.

2.2 Running the Model Reduction Algorithm

The Krylov subspaces allow us to obtain a low-dimensional subspace basis for (2.4) with excellent approximating properties by means of a very efficient computation [11,8]. The current version of mor4ansys implements the block Arnoldi algorithm [11] in order to support multiple inputs, the block size being equal to the number of inputs.

Each step of an iterative Krylov subspace algorithm requires us to compute a matrix-vector product, for example, for the first-order system

$$A^{-1}E\boldsymbol{h} \tag{2.5}$$

where $\boldsymbol{h}$ is some vector. The system matrices are high-dimensional and sparse and one does not compute A^{-1} explicitly. The only feasible solution is to solve a linear system of equations for each step as follows

$$A\boldsymbol{g} = E\boldsymbol{h} \tag{2.6}$$

This constitutes the main computational cost up to the order of the reduced system 30. Later on, the additional cost associated with the orthogonolization process can be also added.

There are many sparse solvers as well as many storage schemes for sparse matrices. Our goal was to implement a model reduction algorithm in a way that does not depend on a particular solver. In addition, we wanted to change solvers at run-time, that is, to allow for run-time polymorphism. As a result, we have chosen the virtual function

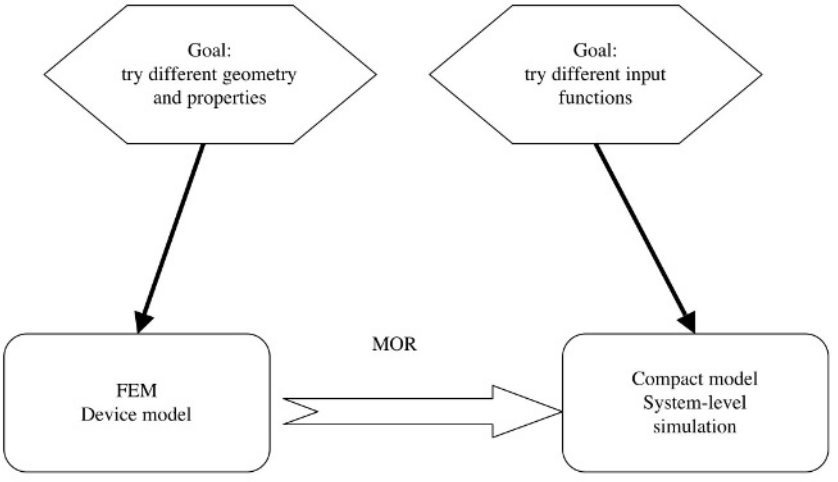

Fig. 2. Use of model reduction during design and system-level simulation

mechanism, as its overhead is negligible in our case when the operations by themselves are computationally intensive.

Our approach is similar to that in the PETs [12] and Trinilos [13] libraries. The abstract interface is written in terms of relatively low-level functions, as the goal was to cover many different scenarios. The vectors are represented by continuous memory, as they are dense in the case of the Krylov subspaces.

At present, the direct solvers from the TAUCS [14] and UMFPACK [15,16] libraries are supported. The ATLAS library [17] has been used to generate the optimized BLAS. We have found that for many ANSYS models up to 500 000 degrees of freedom the modern direct solvers are quite competitive as the matrix factor fits within 4 Gb of RAM. This allows us to reuse the factorization and achieve good performance.

3 Computational Cost of Model Reduction

We have experimentally observed that for many ANSYS models a reduced model of order 30 is enough to accurately represent the original high-dimensional system [5]. Hence, for simplicity we limit the analysis of the computational cost to this case.

The simulation time of the reduced system comprising 30 equations is very small and we can neglect it. Therefore, for the case when several simulations with different input functions are necessary (the system-level simulation case), the advantage of model reduction is out of the question.

Yet, during the design phase, a reduced model should be generated each time when a user changes the geometry or material properties of the original model. In this case, a reduced model might be used just once. Nevertheless, the model reduction time can be smaller than the simulation time of the original system even in this case. These two different situations are shown in Fig. 2. Below we consider the second case.

Table 1. Computational times on Sun Ultra-80 with 4 Gb of RAM in seconds

dimension	nnz	stationary solution in ANSYS 7.0	stationary solution in ANSYS 8.0	factoring in TAUCS	generation of the first 30 vectors
4 267	20 861	0.87	0.63	0.31	0.59
11 445	93 781	2.1	2.2	1.3	2.7
20 360	265 113	16	15	12	14
79 171	2 215 638	304	230	190	120
152 943	5 887 290	130	95	91	120
180 597	7 004 750	180	150	120	160
375 801	15 039 875	590	490	410	420

Let us assume that a direct solver is applicable and the dimension of 30 for the reduced system is sufficient. Then the model reduction time is equal to the time of factoring A in Eq (2.5) and the time required for 30 back substitution steps in Eq (2.6). Table 1 presents computational times for seven ANSYS models where the system matrices are symmetric and positive definite. The first four rows correspond to thermal simulations [18] and the last three to structural mechanics of a bond wire [7].

Each case is specified by its dimension and the number of non zero elements in the stiffness matrix. The time of a stationary solution in ANSYS is given as a reference point. Note that the real simulation time in ANSYS required for the stationary solution is larger than in Table 1 as it includes reading/writing files as well as some other operations. After that is listed the time to factor a matrix by means of a multifrontal solver from the TAUCS library [14] and the time to generate the first 30 vectors. The latter is dominated by the solution of Eq (2.6) by means of back substitution. As the difference to generate the first and thirtieth vectors was less than 10-20%, we can say that the orthogonalization cost was relatively small.

Note that the TAUCS multifrontal solver is even faster than the ANSYS solver. The total time to generate a reduced model is about twice more than that for the stationary solution. At the same time, the reduced model can accurately reproduce any transient and harmonic simulation of the original models within a reasonable frequency range.

The simulation time of a harmonic analysis is the product of solution time for a complex linear system by the number of frequencies needed. The matrix factor cannot be re-used as the linear system to solve depends on frequency. The solution time for a complex linear system is about twice more expensive. Hence model reduction allows us to save simulation time by a factor close to the number of frequencies at which the harmonic response is required. For example, if it is necessary to estimate the transfer function at ten frequencies, then the model reduction plus the simulation of the reduced system is roughly ten times faster than the simulation of the original system.

For the transient simulation, the situation is more difficult to analyse as this depends on the integration strategy. In principle, it is possible to say that the model reduction time above is equivalent to 30 equally spaced timesteps as in this case the same strategy with

the re-use of the matrix factor can be applied. However, in our experience, in order to achieve accurate integration results for the examples in Table 1, one either needs at least 600 equally-spaced timesteps or one needs to use adaptive integration schemes where the factor re-use is not possible. In both cases, model reduction plus simulation of the reduced system was more than ten times faster. This shows that model reduction can also be viewed as a fast solver and can be employed even during the optimization phase.

4 Conclusions

We have shown that in the case of the linear dynamics systems (1.1) and (2.3) modern model reduction techniques can speed up finite element transient and harmonic simulation significantly. For nonlinear systems, there are promising theoretical results in the case of polynomial type nonlinearity [19]. Yet, in the nonlinear case in addition to many theoretical problems, it happens that extracting a nonlinear system (1.1) or (2.3) from a commercial finite element tool is a challenge by itself.

Acknowledgment

ANSYS models of the microthruster and the bonded wire have been made by T. Bechtold and J. Lienemann respectively. We would also like to acknowledge an anonymous reviewer for the very helpful comments and suggestions to improve the paper. Partial funding by the DFG project MST-Compact (KO-1883/6), the Italian research council CNR together with the Italian province of Trento PAT, the European Union (grant EU IST-1999-29047, Micropyros) and an operating grant of the University of Freiburg is gratefully acknowledged.

References

1. A. C. Antoulas, D. C. Sorensen. Approximation of Large-Scale Dynamical Systems: An overview. *Applied Mathematics & Computer Science*, 11(5):1093–1121, 2001.
2. E. B. Rudnyi, J. G. Korvink. Automatic Model Reduction for Transient Simulation of MEMS-based Devices. *Sensors Update*, 11:3–33, 2002.
3. ANSYS, ANSYS Inc. http://www.ansys.com/
4. R. F. Boisvert, R. Pozo, K. A. Remington. The Matrix Market Exchange Formats: Initial Design. *NIST Interim Report 5935*, 1996. http://math.nist.gov/MatrixMarket/
5. E. B. Rudnyi, J. G. Korvink. Model Order Reduction of MEMS for Efficient Computer Aided Design and System Simulation. In *Sixteenth International Symposium on Mathematical Theory of Networks and Systems*, Belgium, July 5-9, 2004. Minisymposium TA8: Issues in model reduction of large-scale systems.
6. E. B. Rudnyi, J. G. Korvink. mor4ansys (version 1.6): Compact Behavioral Models from ANSYS by Means of Model Order Reduction. *User Manual*, 2004. http://www.imtek.uni-freiburg.de/simulation/mor4ansys/
7. E. B. Rudnyi, J. Lienemann, A. Greiner, and J. G. Korvink. mor4ansys: Generating Compact Models Directly from ANSYS Models. In *Technical Proceedings of the 2004 Nanotechnology Conference and Trade Show*, Nanotech 2004, March 7-11, 2004, Bosten, Massachusetts, USA.

8. Z. J. Bai, K. Meerbergen, Y. F. Su. Arnoldi methods for structure-preserving dimension reduction of second-order dynamical systems. In: P. Benner, G. Golub, V. Mehrmann, D. Sorensen (eds), *Dimension Reduction of Large-Scale Systems, Lecture Notes in Computational Science and Engineering.* Springer-Verlag, Berlin/Heidelberg, Germany, 2005.
9. J. G. Korvink, E. B. Rudnyi. Oberwolfach Benchmark Collection. In: P. Benner, G. Golub, V. Mehrmann, D. Sorensen (eds), *Dimension Reduction of Large-Scale Systems, Lecture Notes in Computational Science and Engineering.* Springer-Verlag, Berlin/Heidelberg, Germany, 2005. http://www.imtek.uni-freiburg.de/simulation/benchmark/
10. Guide to Interfacing with ANSYS, ANSYS Inc. 2001.
11. R. W. Freund. Krylov-subspace methods for reduced-order modeling in circuit simulation. *Journal of Computational and Applied Mathematics*, 123: 395–421, 2000.
12. S. Balay, V. Eijkhout, W. D. Gropp, L. C. McInnes, B. F. Smith. Efficient Management of Parallelism in Object Oriented Numerical Software Libraries. In E. Arge, A. M. Bruaset, H. P. Langtangen (eds), *Modern Software Tools in Scientific Computing*, Birkhäuser Press, 163–202, 1997.
http://www-unix.mcs.anl.gov/petsc/petsc-2/
13. M. Heroux, R. Bartlett, V. Howle, R. Hoekstra, J. Hu, et al. An Overview of Trilinos. *Sandia National Laboratories report* SAND2003-2927, 2003. http://software.sandia.gov/trilinos/
14. V. Rotkin, S. Toledo. The design and implementation of a new out-of-core sparse Cholesky factorization method. *ACM Transactions on Mathematical Software*, 30: 19–46, 2004.
http://www.tau.ac.il/ stoledo/taucs/
15. T. A. Davis. Algorithm 832: UMFPACK V4.3, an unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software*, 30(2): 196–199, 2004.
http://www.cise.ufl.edu/research/sparse/umfpack/
16. T. A. Davis. A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software*, 30(2): 165–195, 2004.
17. R. C. Whaley, A. Petitet, J. Dongarra. Automated Empirical Optimization of Software and the ATLAS project. *Parallel Computing*, 27(1-2): 3-35, 2001.
http://math-atlas.sourceforge.net/
18. J. Lienemann, E. B. Rudnyi, J. G. Korvink. MST MEMS model order reduction: Requirements and Benchmarks. Submitted to *Linear Algebra and its Applications*, 2004.
19. J. R. Phillips. Projection-based approaches for model reduction of weakly nonlinear, time-varying systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22:171–187, 2003.