ARCHIVE OF MECHANICAL ENGINEERING

VOL. LXIII 2016 Number 4

DOI: 10.1515/meceng-2016-0027

Key words: model order reduction, elastic multibody systems, high performance computing, parallel reduction, Trilinos, Anasazi, iterative solver

THOMAS VOLZER[1], *PETER EBERHARD*[1]

MODEL ORDER REDUCTION OF LARGE-SCALE FINITE ELEMENT SYSTEMS IN AN MPI PARALLELIZED ENVIRONMENT FOR USAGE IN MULTIBODY SIMULATION

The use of elastic bodies within a multibody simulation became more and more important within the last years. To include the elastic bodies, described as a finite element model in multibody simulations, the dimension of the system of ordinary differential equations must be reduced by projection. For this purpose, in this work, the modal reduction method, a component mode synthesis based method and a moment-matching method are used. Due to the always increasing size of the non-reduced systems, the calculation of the projection matrix leads to a large demand of computational resources and cannot be done on usual serial computers with available memory. In this paper, the model reduction software Morembs++ is presented using a parallelization concept based on the message passing interface to satisfy the need of memory and reduce the runtime of the model reduction process. Additionally, the behaviour of the Block-Krylov-Schur eigensolver, implemented in the Anasazi package of the Trilinos project, is analysed with regard to the choice of the size of the Krylov base, the blocksize and the number of blocks. Besides, an iterative solver is considered within the CMS-based method.

1. Introduction

Lately, the use of elastic bodies within multibody simulations (MBS) found increasing application in scientific and industrial projects. The elastic bodies described as finite element (FE) models result in a system of ordinary differential equations (ODE). The dimension of these ODE became huge due to ever increasing refinements of models and simulation quality demands. The tremendous development of the computational resources with respect to the amount of memory and

[1] *Institute of Engineering and Computational Mechanics, University of Stuttgart, Pfaffenwaldring 9, 70569 Stuttgart, Germany. Emails: thomas.volzer@itm.uni-stuttgart.de, peter.eberhard@itm.uni-stuttgart.de*

the speed of the central processing unit (CPU) made such demanding applications possible but are still restricting a general use. Before these ODE can be included in an MBS, their dimension must be reduced to a much smaller size which is accomplished by the projection-based model reduction. Thereby, the central issue is the calculation of a projection matrix by the use of different reduction methods. In this work, the modal reduction method, a component mode synthesis (CMS) based method, and a moment-matching method [1] are used to calculate the projection matrix. While the first two mentioned method classes are well established in industry and science, the latter has been a subject of research in science just for the past years. Within each reduction method, the most expensive step, with respect to the need of memory as well as to the runtime, is often solving a sparse system of linear equations. With the projection matrix, a reduced system of linear equations can be calculated and included into the MBS according to the theory described in [2, 3].

At the Institute of Engineering and Computational Mechanics (ITM) the two program systems Morembs++ and MatMorembs [4] have been developed within the last years. While MatMorembs is written in the programming language Matlab and aims primarily at small to medium sized FE models, Morembs++ is specifically developed with regard to large scale FE models and written in the programming language C++. To mention typical numbers, MatMorembs can usually be used for systems with several millions degrees of freedom (DOF) while Morembs++ must certainly be used once 5 million DOF or more must be considered. MatMorembs contains also more specific model order reduction methods, like parametric model reduction techniques [5]. Morembs++ functions as a preprocessor tool between the FEM and the MBS software. The process chain is divided into an import from the FE data, the reduction in Morembs++ and the export into a data format used by the external MBS software. Additionally, the transfer function of the full-sized and reduced model as well as the relative error between both models can be calculated in Morembs++. Morembs++ is used on a single, usual workstation, on multiple workstations, connected in a local area network (LAN) as well as on cluster and supercomputers. To make use of distributed systems, Morembs++ is parallelized based on the message passing interface (MPI) and makes heavily use of libraries from the Trilinos project [6]. The Trilinos project is divided into packages, each with different applications. One of these packages is named Anasazi and contains an implementation of the Block-Krylov-Schur (BKS) eigensolver [7]. Except for the CMS based method, only direct methods can currently be used in Morembs++ due to the properties of the coefficient matrix.

When dealing with large-scale finite element systems in a parallelized environment, domain decomposition methods are often used [8]. Thereby, the global domain is separated into smaller domains which are then treated independently. These independent subdomains can be assigned to different processes in an MPI-parallelized system. By the way the interfaces between these separated domains are defined, domain decomposition methods are categorized in overlapping and non-overlapping methods [9]. The latter can be classified further into balancing domain decomposition (BDD) methods [10] and finite element tearing and interconnect (FETI) methods [11] depending on the description of the interface domain. In [8] non-overlapping methods in the context of structural mechanics are considered.

As a preconditioner, the use of domain decompositioning can be easily included into Morembs++ as part of the iterative solution process. Within the Ifpack2 [12] package of the Trilinos project overlapping and non-overlapping Additive-Schwarz methods are available. On a subdomain level different solution methods can be used.

In an elastic multibody system many different elastic bodies are interacting only on the interface DOF. This substructuring also can be interpreted as a decomposition method in a mechanical sense. Domain decomposition methods from finite elements as described above also can be used on a substructural level. However, this is not considered here because in this contribution the focus is on the memory consumption during the reduction within one substructure.

The novel contribution of this work is the investigation of model order reduction of large-scale FE models using the alternative reduction method moment-matching in an MPI-based, high performance computing (HPC) environment.

In this work, three different sized FE models are imported from the FE software PERMAS [13] and reduced in Morembs++. Thereby, the computational resources needed for the calculation of the projection matrix are examined and the results of the different reduction methods are compared to each other. The reduction process is carried out on single computers as well as on distributed systems within a LAN and on the Cray supercomputer at the High-Performance Computing Center Stuttgart (HLRS) where the scalability of the different parts of the model reduction is examined in detail.

Also, the BKS eigensolver is examined with regard to the size of the Krylov base, the blocksize, the number of blocks and the runtime. Besides, the usage of iterative solvers is examined within the CMS based method.

2. Background

The system of second order ordinary differential equations

$$\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{K}\boldsymbol{q} = \boldsymbol{B}\boldsymbol{u} \tag{1}$$

with the mass matrix $\boldsymbol{M} \in \mathbb{R}^{N\times N}$, the stiffness matrix $\boldsymbol{K} \in \mathbb{R}^{N\times N}$, the input matrix $\boldsymbol{B} \in \mathbb{R}^{N\times p}$, the vector $\boldsymbol{q} \in \mathbb{R}^{N}$ of elastic DOF, the number N of all DOF and the number p of input DOF describes the FE model of an elastic body. Before the inclusion into the MBS can happen, the dimension N of the matrices in Equation (1) must be reduced by the approximation

$$\boldsymbol{q} \approx \boldsymbol{V}\tilde{\boldsymbol{q}} \tag{2}$$

with the projection matrix $\boldsymbol{V} \in \mathbb{R}^{N\times n}$, the vector of the reduced elastic DOF $\tilde{\boldsymbol{q}} \in \mathbb{R}^{n}$ and the dimension n of the reduced elastic system. In this work, the modal reduction method, a CMS based method and a moment-matching method are used to calculate the projection matrix $\boldsymbol{V}$. The usage of a moment-matching based model order reduction method has been shown in [14]. A possible combination of a CMS and a moment-matching based method has also been presented in [15].

The modal reduction method mainly involves solving the generalized eigenvalue problem

$$\boldsymbol{K}\boldsymbol{\phi}_i = \omega_i^2 \boldsymbol{M}\boldsymbol{\phi}_i \tag{3}$$

with the eigenvector $\boldsymbol{\phi}_i$, the angular eigenfrequency $\omega_i = 2\pi f_i$ and the eigenfrequency f_i of the undamped system. The BKS eigensolver, implemented in the package Anasazi of the Trilinos project, is only capable of solving the simple eigenvalue problem. Therefore, the transformation of the generalized eigenvalue problem into the simple eigenvalue problem is done using the shift-and-invert strategy [16]

$$\frac{1}{\omega_i^2 - \sigma}\boldsymbol{\phi}_i = \boldsymbol{A}^{-1}\boldsymbol{M}\boldsymbol{\phi}_i \tag{4}$$

with the matrix $\boldsymbol{A} = \boldsymbol{K} - \sigma\boldsymbol{M}$ and the shift parameter σ. The number of eigenvectors, respectively eigenvalues, to be calculated depends on the dimension n of the reduced system which leads to the projection matrix

$$\boldsymbol{V} = [\boldsymbol{\phi}_1 \cdots \boldsymbol{\phi}_n]. \tag{5}$$

The calculation of a Krylov base of the form

$$\mathcal{K} = \left\{\boldsymbol{V}_0, (\boldsymbol{A}^{-1}\boldsymbol{M})\boldsymbol{V}_0, \ldots, (\boldsymbol{A}^{-1}\boldsymbol{M})^{n_\mathrm{K}-1}\boldsymbol{V}_0\right\}, \tag{6}$$

with the size n_K of the Krylov base, the normalized starting vector $\boldsymbol{V}_0 \in \mathbb{R}^{N\times n_\mathrm{BS}}$ and the blocksize n_BS is the most expensive part while solving the eigenvalue problem in Morembs++. The blocksize determines how many base vectors are computed in each iteration while the number n_B of blocks stands for the number of iterations. Both, the blocksize and the number of blocks define the size of the Krylov base by $n_\mathrm{K} = n_\mathrm{BS}\, n_\mathrm{B}$. In each iteration, a sparse system of linear equations with the coefficient matrix $\boldsymbol{A}$ has to be solved for each newly calculated base vector. The frequency shift parameter σ defines the part of the wanted spectrum and usually leads to an indefinite coefficient matrix. The choice of the blocksize and the number of blocks are discussed in this work.

The CMS method used here is based on the substructuring

$$\boldsymbol{M}_\mathrm{ii} = \boldsymbol{T}_\mathrm{i}^\top \boldsymbol{M}\boldsymbol{T}_\mathrm{i}, \tag{7}$$

$$\boldsymbol{K}_\mathrm{ii} = \boldsymbol{T}_\mathrm{i}^\top \boldsymbol{K}\boldsymbol{T}_\mathrm{i}, \tag{8}$$

$$\boldsymbol{K}_\mathrm{ib} = \boldsymbol{T}_\mathrm{i}^\top \boldsymbol{K}\boldsymbol{T}_\mathrm{b} \tag{9}$$

described in [17], where the index i describes the inner and the index b the bounded DOF. The matrices $\boldsymbol{M}_{\mathrm{ii}}$, $\boldsymbol{K}_{\mathrm{ii}}$ and $\boldsymbol{K}_{\mathrm{ib}}$ represent submatrices of the appropriate system matrices while the transformation matrices $\boldsymbol{T}_{\mathrm{i}}$ and $\boldsymbol{T}_{\mathrm{b}}$ are used to calculate these submatrices. The constraint modes can be calculated by

$$\boldsymbol{\Psi}_{\mathrm{c}} = \begin{bmatrix} -\boldsymbol{K}_{\mathrm{ii}}^{-1}\boldsymbol{K}_{\mathrm{ib}} \\ \boldsymbol{I}_{\mathrm{bb}} \end{bmatrix}, \tag{10}$$

where $\boldsymbol{I}_{\mathrm{bb}}$ is the $n_{\mathrm{b}} \times n_{\mathrm{b}}$ identity matrix and n_{b} the number of bounded DOF. The normal modes $\boldsymbol{\phi}_{\mathrm{n},i}$ of the undamped inner system are the solution of the generalized eigenvalue problem

$$\boldsymbol{K}_{\mathrm{ii}}\boldsymbol{\phi}_{\mathrm{n},i} = \omega_i^2 \boldsymbol{M}_{\mathrm{ii}}\boldsymbol{\phi}_{\mathrm{n},i} \tag{11}$$

and summarized in

$$\boldsymbol{\Phi}_{\mathrm{n}} = \begin{bmatrix} \boldsymbol{\phi}_{\mathrm{n},1} & \cdots & \boldsymbol{\phi}_{\mathrm{n},n_{\mathrm{i}}} \\ \boldsymbol{0} & \cdots & \boldsymbol{0} \end{bmatrix} \tag{12}$$

with the number n_{i} of inner eigenmodes. The projection matrix

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{\Psi}_{\mathrm{c}} & \boldsymbol{\Phi}_{\mathrm{n}} \end{bmatrix} \tag{13}$$

consists of both the constraint modes and the normal modes of the inner system. Within the CMS method, two sparse systems of linear equations with the coefficient matrices $\boldsymbol{K}_{\mathrm{ii}}$ and $\boldsymbol{K}_{\mathrm{ii}} - \sigma\boldsymbol{M}_{\mathrm{ii}}$ have to be solved. While the first coefficient matrix is part of the calculation of the constraint modes, the second one is used within the BKS eigensolver when normal modes are needed. Both the inner mass matrix $\boldsymbol{M}_{\mathrm{ii}}$ and the inner stiffness matrix $\boldsymbol{K}_{\mathrm{ii}}$ are positive definite. Therefore, even if the coefficient matrix $\boldsymbol{K}_{\mathrm{ii}}$ is positive definite, the coefficient matrix $\boldsymbol{K}_{\mathrm{ii}} - \sigma\boldsymbol{M}_{\mathrm{ii}}$ can be indefinite depending on the frequency shift parameter σ.

The moment-matching method is based on the approximation of the transfer function

$$\boldsymbol{G} = \boldsymbol{B}^{\top}\left(\boldsymbol{K} - 4\pi^2 f_i^2 \boldsymbol{M}\right)^{-1}\boldsymbol{B}. \tag{14}$$

Thereby, the values and optionally higher-order derivatives of the moments of this function are matched at specific frequency points f_i. The input matrix $\boldsymbol{B}$ depends on the interface DOF which are usually the points where the elastic body is included into the MBS. The number of columns of the projection matrix $\boldsymbol{V}$ increases with the number of frequency points and whether higher-order derivatives are considered or not. The calculation of the projection matrix $\boldsymbol{V}$ is done mainly by the Arnoldi algorithm [14]. Similar to the solution of the generalized eigenvalue problem within the BKS solver, a sparse system of linear equations with the indefinite coefficient matrix $\boldsymbol{A}$ has to be solved at each frequency point f_i and optionally multiple times if higher-order derivatives are desired.

With the calculated projection matrix $\boldsymbol{V}$ the equation of motion of the reduced elastic body can be described by

$$\tilde{\boldsymbol{M}}\ddot{\tilde{\boldsymbol{q}}} + \tilde{\boldsymbol{K}}\tilde{\boldsymbol{q}} = \tilde{\boldsymbol{B}}\boldsymbol{u} \tag{15}$$

with the reduced mass matrix $\tilde{\boldsymbol{M}} = \boldsymbol{V}^\top \boldsymbol{M}\boldsymbol{V}$, the reduced stiffness matrix $\tilde{\boldsymbol{K}} = \boldsymbol{V}^\top \boldsymbol{K}\boldsymbol{V}$ as well as the reduced input matrix $\tilde{\boldsymbol{B}} = \boldsymbol{V}^\top \boldsymbol{B}$. These reduced systems are then included into the multibody formulation.

3. Process Chain

The model reduction software Morembs++ has been written in the programming language C++ and provided specifically to treat large-scale FE models. In Fig. 1 the process chain of Morembs++ is shown. All model reduction settings are given within an XML configuration file. Morembs++ is separated into four different parts. First, the FE data must be imported from the FEM software and the results are saved in an HDF5 file [18]. Afterwards, the model reduction process can be executed and the resulting projection matrix $\boldsymbol{V}$ as well as the reduced system matrices $\tilde{\boldsymbol{M}}$, $\tilde{\boldsymbol{K}}$, and $\tilde{\boldsymbol{B}}$ are saved in a separate HDF5 file. The last step is the preparation of the data of the reduced elastic body, so that it can be included in the MBS simulation. The transfer function calculator (TFC) is not directly connected to the model reduction process but rather an optional step to estimate the quality of the reduced elastic body. Within the transfer function calculator, the transfer function

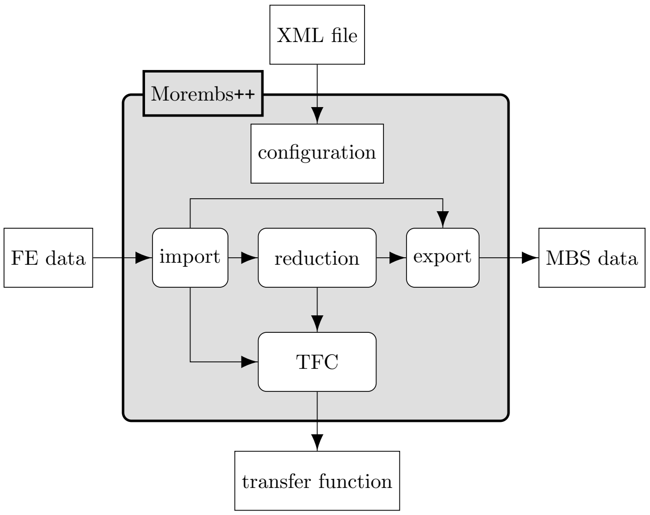

Fig. 1. Process chain of the model order reduction software Morembs++

of the unreduced elastic body, of the reduced body, and the relative error between both can be calculated. The results are saved in a separated HDF5 file, too.

Due to the ill-conditioned system matrices and the possible indefiniteness of the coefficient matrix, in general only direct solvers can be used in Morembs++ currently. There are many libraries available, in which different sparse direct solvers have been implemented. During the development of Morembs++ the MUMPS [19], SuperLU_Dist [20] and the Pardiso [21] solver were considered for application. At the time of the development of Morembs++ the Pardiso solver did not support MPI-based parallelism and the SuperLU_Dist solver lacked an out-of-core (OOC) functionality. Therefore, the direct solver implemented in the library MUMPS is applied in Morembs++. Within the calculation of the constraint modes, the sparse coefficient matrix of the system of linear equations to be solved is positive definite and can be solved iteratively using the preconditioned conjugate gradient (CG) method. To satisfy the huge demands of memory, Morembs++ makes use of MPI-based parallelism. Thereby, the data is distributed to different computers to obtain the agglomerated amount of memory of all participating computers. To simplify the development process, Morembs++ makes heavy use of the Trilinos project which is separated in different packages each focusing on a specific topic. The Tpetra package, for instance, contains linear algebra objects while the Teuchos package offers software tools like smart pointers or an XML parser. Within the Anasazi package different eigensolvers are implemented where one of them is the already mentioned BKS eigensolver.

4. Numerical Examples

Three different sized FE models are examined here. The first is a model of a crankshaft, the second FE model originates from an automotive gear and the third FE model describes a complete carriage of an automotive. Some key numbers of these FE models being important to the model order reduction process, are given in Table 1. These are the dimension n_{nodes} of nodes, the dimension n_{els} of elements, the number N of DOF of the unreduced model, the number n_{nz} of non-zero elements of the sparse coefficient matrix $\boldsymbol{A}$ and the number p of input and output DOF.

Additionally, in Fig. 2 the FE models of the crankshaft, the gear and the carriage are shown. The white marks show the interface nodes at which the models are included in the flexible multibody system. The crankshaft has four the gear 29,

Table 1.
Important information of the examined FE models related to the model order reduction in Morembs++

	n_{nodes}	n_{els}	N	n_{nz}	p
Crankshaft	158 782	101 542	444 837	34 841 153	24
Gear	1 336 209	837 027	3 988 122	332 930 880	156
Carriage	2 039 638	1 804 778	9 242 089	715 664 213	246

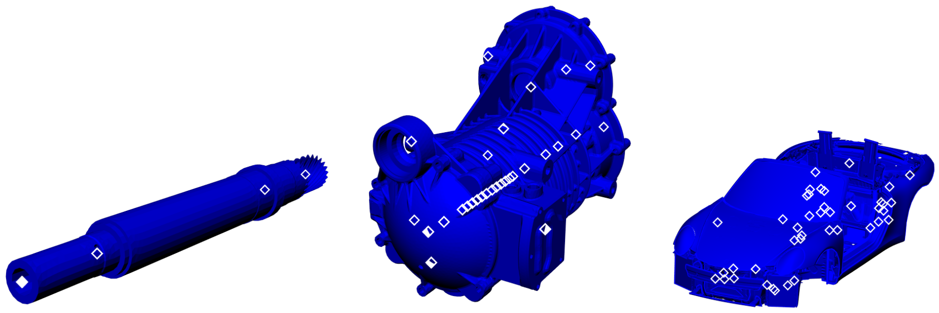

Fig. 2. Figures of the FE model of the crankshaft, the gear and the carriage. The white marks show the interface nodes at which the models are included into the flexible multibody system

and the carriage 41 interface nodes. Each of this interface nodes owns six DOF which lead to the number of interface DOF shown in Table 1.

Independent of the reduction method used, the solution of a sparse system of linear equations is the most expensive part regarding the consumption of memory. The worst case scenario, concerning the memory consumption, occurs when the sparse coefficient matrix $\boldsymbol{A}$ is indefinite which is true when the transfer function at an arbitrary frequency point greater than zero is calculated. Because of this, the calculation of the transfer function at one frequency point is representative for all reduction methods with respect to the consumption of memory.

In Table 2, the computer systems used in this work are listed. It is assumed that usually multiple personal computers (type I) are available while only one workstation (type II) can be used at a time. Here, these systems are chosen to describe the methods in conjunction with the model size in the best way. Of course, there could be used computers with even more memory but with a further increasing size of the FE models the same type of limits would occur, only at a higher level. The Cray supercomputer at the HLRS (type III) consists of 7712 compute nodes where each one of them has two CPUs and 128 GB of memory. This leads to 15424 CPUs and an agglomerated size of memory of 964 TB.

Table 2.

Computer systems used here

type	description	CPU	RAM
I	PC	Core i7-3770	32 GB
II	workstation	Xeon E5-1650 v3	128 GB
III	Cray HLRS	Xeon E5-2680 v3	128 GB

In Fig. 3 the memory consumption during the calculation of the transfer function of the crankshaft at one frequency point is shown on the left. For better comparison of different runs, the time is scaled from zero to one. At first, the memory consumption increases due to the import of the system matrices $\boldsymbol{M}$ and

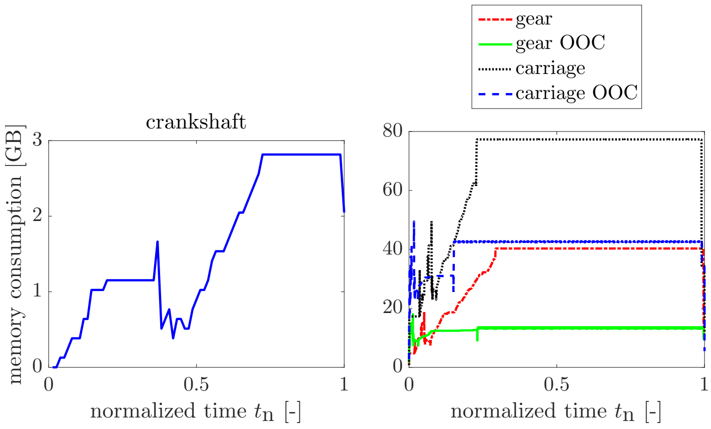

Fig. 3. Memory consumption during the solution of the sparse system of linear equalities

K which is followed by the calculation of the coefficient matrix ***A***. Afterwards, the memory consumed by ***M*** and ***K*** can be freed which becomes visible at the decline of the memory consumption left of the 0.4 time mark. The coefficient matrix must be transformed to the format used within the direct solver MUMPS [19] and an ordering is necessary to minimize the memory consumption during the numerical factorization. These operations lead to a fluctuation of the memory consumption shortly before and after the 0.4 time mark. The subsequently executed numerical factorization consumes the largest part of memory during the solution of the sparse system of linear equations. This procedure ends shortly after the 0.7 time mark and is followed by the forward and backward substitution process which is characterized by the constant consumption of memory until the end of the shown timeframe. The maximum memory consumption of a little less than 3 GB is reached after the numerical factorization.

In Fig. 3, the memory consumption during the calculation of the transfer function of the gear and the carriage at one frequency point are shown on the right. Again, the time is scaled from zero to one for better comparison of the different runs. The sectors can be classified in an equal way as it is done above with the crankshaft. The memory consumption using the gear is shown as a dashdotted graph in Fig. 3. The maximum consumption of memory is approximately 40 GB which is more than the capacity of a computer of type I. One solution is to use a system of type II but the maximum capacity of memory is always limited due to technological standards and so this is just pushing the limits a bit further. Another possibility is to use the out-of-core (OOC) functionality of the direct solver MUMPS. Thereby, during the numerical factorization, a part of the accumulated data is written to the hard disk and imported again when necessary as part of the forward and backward substitution process. Due to this approach, the maximum memory consumption drops to approximately 10 GB which is shown by the solid line in Fig. 3 and a type I system can be used.

Considering the FE model of the carriage the serial, in-core solution process leads to a maximum consumption of memory of approximately 80 GB which is shown by the dotted graph in Fig. 3. By using the OOC functionality, the maximum memory consumption can be reduced to about 40 GB as it can be seen by the dashed graph in Fig. 3 which still exceeds the memory limit of a type I computer. To summarize the result, the OOC functionality can reduce the maximum amount of memory needed during the solution of the sparse system of linear equations, but with an increasing size of the FE model, a type I computer with 32 GB of memory cannot be used anymore. Instead, a stronger equipped computer, regarding the amount of available memory, is necessary. Even if such a computer is available, the limit can be reached quickly when even larger FE models are required.

Another approach to solve this memory related issue is to distribute the incurred data to multiple computers using MPI-based parallelism. In Fig. 4 the upper left plot shows the memory consumption during the calculation of the transfer function of the FE model of the gear at one frequency point. Again, the dotted graph shows the memory consumption in the serial case while the two solid lines show the memory consumption on each MPI process when two MPI processes are used. Instead of 40 GB of maximum memory consumption, each MPI process needs approximately 20 GB. Given that each MPI process is assigned to a different computer with a memory capacity of 32 GB, this approach makes it possible to use usual multiple type I computers.

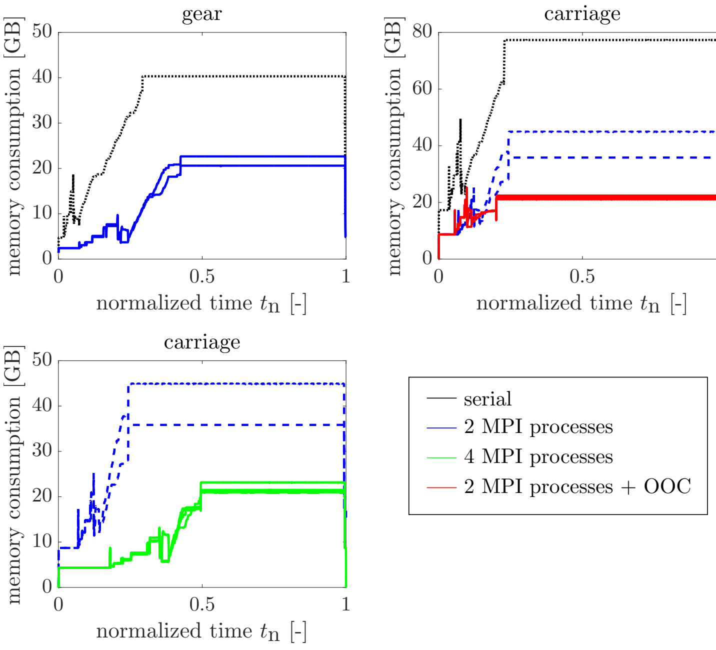

Fig. 4. Memory consumption using the FE models of the gear and the carriage in an MPI-parallelized environment

In Fig. 4, the upper right plot shows the memory consumption during the calculation of the transfer function of the FE model of the carriage at one frequency point in the serial as well as in the MPI-parallelized case. The dotted drawn graph shows the memory consumption in the serial case while the dashed graphs show the memory consumption using two MPI processes. Each MPI process consumes between 38 GB and 45 GB of memory which still exceeds the limit of 32 GB. There are two possibilities to decrease the memory needed per MPI process. Either more MPI processes, and with it more computers, are used to increase the agglomerated amount of memory or the OOC functionality is used on each MPI process. In Fig. 4 the solid graph in the upper right plot shows the latter case when the OOC functionality is activated. The memory consumption per MPI process drops to approximately 20 GB, so that two type I computers can be used again. The solid line in the lower left plot in Fig. 4 shows the case when four MPI processes are used. Equally to the case when the OOC functionality is applied, the maximum amount of memory is about 20 GB and, therefore, four type I computers are needed to perform the calculation of the transfer function in-core.

The conclusion of the examination of the maximum needed memory in the context of the model reduction of large FE systems is that the large amount of memory during the solution of a sparse linear system of equations can be faced by three approaches. One possibility is to increase the amount of available memory due to hardware upgrades but of course this possibility is limited. Another way is to use the OOC functionality to reduce the memory consumption, which works but only moves the limit temporarily until a larger model is used. The third option is to use MPI-based parallelism, optionally in conjunction with the OOC functionality. Here, the amount of memory needed is distributed to different computers and the agglomerated amount of memory depends on the number of participating computers. By doing this, the available memory can be made very large and is sufficient in the context of the model reduction, even if very large models are used.

So far, only the memory consumption during the model reduction has been considered. Besides that, the runtime is also a very important issue. In Fig. 5 the most time consuming parts during the calculation of the transfer function at one frequency point are shown. These are the runtime needed to import the data of the system matrices $\boldsymbol{M}$, $\boldsymbol{K}$ and the RHS $\boldsymbol{B}$ as well as the duration of the symbolic and numeric factorization and the forward and backward substitution. The runtimes, shown in Fig. 5, are normalized to the complete runtime of Morembs++ to allow a better comparison between the different runs. When the FE model of the crankshaft is used, the most time consuming parts are the numerical factorization and the forward and backward substitution with both around 40 % percent of the complete runtime. The HDF5 import needs approximately 2 % and the symbolic factorization roughly 7 %.

Using the FE model of the gear, the relative runtimes are different. With around 72 %, the forward and backward substitution is clearly the most time consuming part, followed by the numerical factorization with 22 %. The HDF5 import and

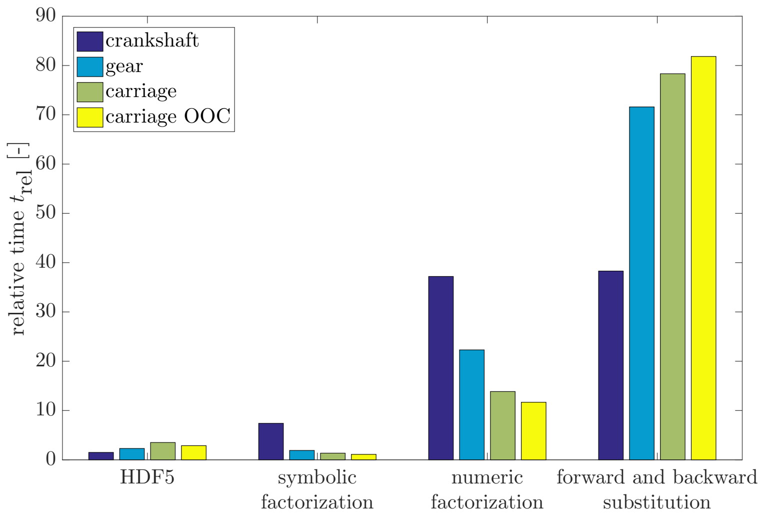

Fig. 5. Most time consuming parts during the calculation of the transfer function at one frequency point

the symbolic factorization are quite small with each below 3 %. Based on the FE model of the carriage, the relative runtime needed for the forward and backward substitution makes up nearby 80 % of the complete runtime while the relative runtime for the numerical factorization is around 14 %. Both the relative runtime needed for the HDF5 import and the symbolic factorization are below 4 %. The increasing relative time of the forward and backward substitution can be traced back to the increasing number p of input DOF leading to the number of RHS to solve. The value of p of the different FE models of the crankshaft, the gear and the carriage is shown in Table 1. The conclusion of this examination is, that depending on the number of RHS, either the numerical factorization or the forward and backward substitution makes up the most time consuming part during the calculation of the transfer function at one frequency point considering only serial runs.

Additionally, the relative runtimes with an activated OOC functionality using the FE model of the carriage are shown in Fig. 5. With around 82 %, the relative runtime needed for the forward and backward substitution process is slightly higher due to the activation of the OOC functionality. In contrast, the relative runtime of the numerical factorization is reduced by 2 % to approximately 12 %. The relative runtimes of the remaining operations are all below 4 %.

The absolute runtimes for the calculations described in Fig. 5 are 54 s, 39 min, 74 min and 91 min based on the FE models of the crankshaft, the gear, and the carriage. These numbers show that the activation of the OOC functionality leads to an increase of the runtime of approximately 22 %. While the primary reason to parallelize Morembs++ using MPI is to obtain a very large capacity of agglomerated memory, the reduction of the runtime is also of great interest. Therefore, the scalability is examined using the FE model of the carriage on the supercomputer

Hazelhen at the HLRS. In Fig. 6 the runtimes of the most time consuming parts during the calculation of the transfer function at one frequency point are shown as a function of the number of MPI processes.

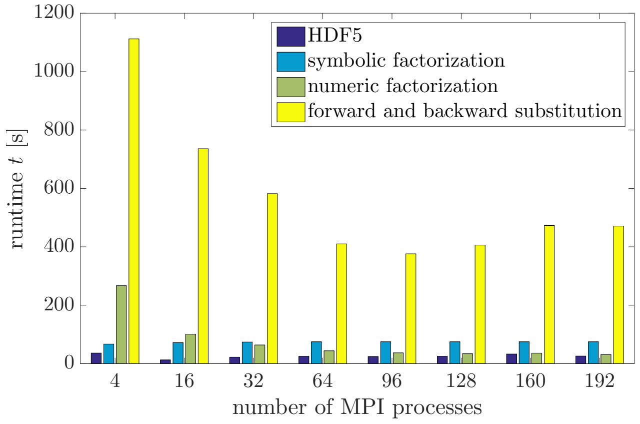

Fig. 6. Scalability of the most time consuming parts during the calculation of the transfer function at one frequency point

The runtimes of the HDF5 input and output (IO) and of the symbolic factorization form only a small part of the complete runtime. Between 4 and 16 MPI processes the runtime of the HDF5 IO drops but increases afterwards. The runtime of the symbolic factorization is independent of the number of processes used. This is because the ordering is done by applying the non-parallelized METIS library [22]. There are MPI parallelized libraries available, like ParMETIS [22], but the runtime of the symbolic factorization is relatively small, compared to the runtime of the numerical factorization as well as to the forward and backward substitution. The runtime of the numerical factorization depends strongly on the number of MPI processes used which is shown in Fig. 6. With four MPI processes, the numerical factorization needs 267 s while using 192 MPI processes it takes only 31 s. However, the runtimes of the forward and backward substitution behave differently. The bars in Fig. 6 show that using more than 96 MPI processes even increase the runtime. The conclusion of this examination is that there is a good scalability of both the numerical factorization as well as the forward and backward substitution process until roughly 96 MPI processes. This is an important result because, as it is shown in Fig. 5, these are the most time consuming parts during the examined calculations. The presented results of the calculation of the transfer function at one frequency point can be transferred to all model reduction methods because the solution of the sparse system of linear equations is always the dominant operation.

In Fig. 7, the transfer functions of differently reduced models are shown in a frequency range between 55 Hz and 65 Hz. The transfer function based on the modal reduction method is shown as a solid line and the peaks describe the eigenfrequencies of the free, undamped system. The transfer function based on the CMS

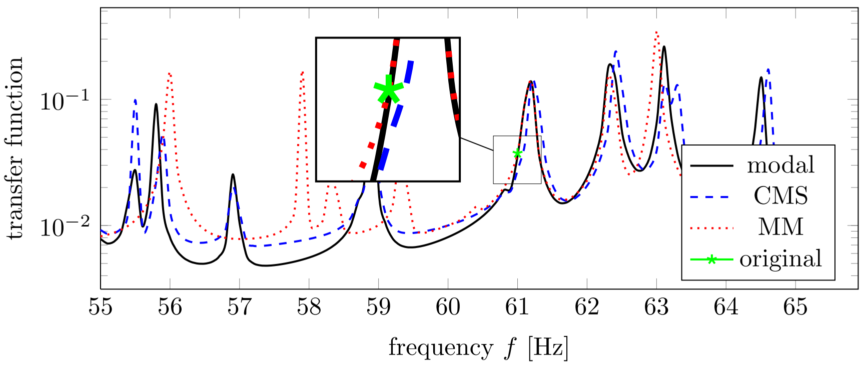

Fig. 7. Transfer functions of the reduced systems using the modal reduction method, the CMS-based method, the moment-matching method, and the original, non-reduced system at one frequency point

reduction method is drawn as a dashed line and shows a similar behaviour within the complete frequency range as the solid graph. The slight difference appears because, due to the substructuring within the CMS reduction method, the dynamics of the inner system is altered. The transfer function drawn as a dotted line originates from a model reduced with the moment-matching method. Thereby, the moments of the transfer function are matched at 61 Hz and no higher-order derivatives are matched. Outside the frequency range around 61 Hz there are big differences to the transfer functions reduced with the other reduction methods. Additionally, the transfer function of the unreduced system is calculated at a frequency point of 61 Hz and marked by a star in Fig. 7. As expected, it can be seen that the moment-matching based transfer function matches the value of the frequency point of 61 Hz.

The BKS eigensolver is an important part of the modal reduction method and the CMS-based method. There are four parameters controlling the eigenvalue solution process. These are the convergence tolerance, the number of eigenvalues to be calculated, the blocksize and the number of blocks. Examinations have shown that a convergence tolerance of $\tau = 10^{-5}$ leads to a sufficient accuracy here. The number of eigenvalues is given by the user and directly affects the blocksize as well as the number of blocks. Both, the blocksize and the number of blocks define the size of the Krylov base by $n_\mathrm{K} = n_\mathrm{BS}\, n_\mathrm{B}$. Initially, the blocksize is set $n_\mathrm{BS} = 1$ to examine the necessary size of the Krylov base. Within the Anasazi package of the Trilinos project, the default size of the Krylov base is $n_\mathrm{K} = 3\, n_\mathrm{EV}$ assuming a blocksize of $n_\mathrm{BS} = 1$. By reducing the size of the Krylov base successively, the convergence behaviour is examined. In the following, the number of times a linear system of equalities has to be solved during the eigenvalue computation is named by number n_sol of solutions. In Fig. 8 the influence of the size of the Krylov base on the number of solutions and on the number of times the iteration has to be restarted during the eigenvalue computation of the FE model of the crankshaft is shown.

At first, the computation of $n_\mathrm{EV} = 40$ eigenvalues is considered. The reduction of the size of the Krylov base leads to a reduction of the number of solutions until

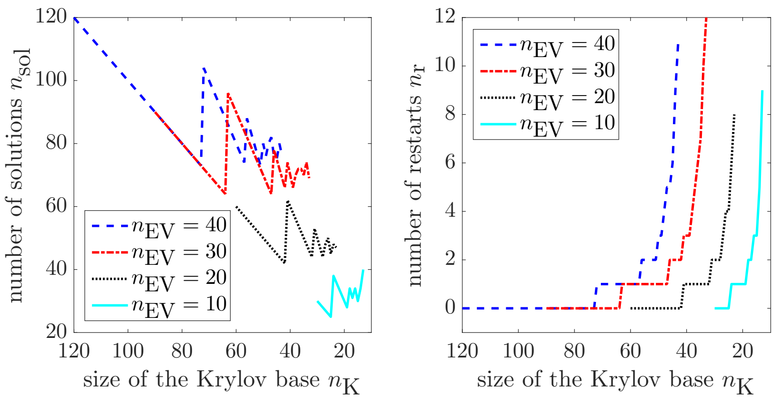

Fig. 8. Influence of the size of the Krylov base on the number of solutions and on the number of times the iterations has to be restarted during the calculation of the eigenvalues of the FE model of the crank

a Krylov base of size $n_{\mathrm{K}} = 72$ enforces a restart of the iteration. Restarting the iteration increases the number of solutions instantly from $n_{\mathrm{sol}} = 73$ to $n_{\mathrm{sol}} = 104$. By further reducing the size of the Krylov base, the number of solutions decreases coming from $n_{\mathrm{sol}} = 104$. This continues until a Krylov base of size $n_{\mathrm{K}} = 56$, when a second restart is necessary. Again, the number of solutions increases instantly from $n_{\mathrm{sol}} = 74$ to $n_{\mathrm{sol}} = 88$. Even smaller sizes of Krylov bases lead to more restarts, which is shown on the right in Fig. 8. Within the computations shown here, the optimal size of the Krylov base is the lowest possible which doesn't lead to a restart. Depending on the number of wanted eigenvalues, the optimal sizes of the Krylov bases are $n_{\mathrm{K}}(n_{\mathrm{EV}} = 40) = 73$, $n_{\mathrm{K}}(n_{\mathrm{EV}} = 30) = 64$, $n_{\mathrm{K}}(n_{\mathrm{EV}} = 20) = 42$ and $n_{\mathrm{K}}(n_{\mathrm{EV}} = 10) = 25$. The optimal size of the Krylov base depends on multiple parameters, like the condition of the system matrices, the convergence tolerance and the number of eigenvalues, and cannot be determined before the computation is done.

The blocksize is examined based on the computation of $n_{\mathrm{EV}} = 40$ eigenvalues of the FE models of the crankshaft and of $n_{\mathrm{EV}} = 50$ eigenvalues of the FE model of the gear. Considering the FE models of the crankshaft, a constant size of the Krylov base of $n_{\mathrm{K}} = 80$ is used. In Table 3 all possible combinations of the blocksize and the number of blocks which lead to a size of the Krylov base of $n_{\mathrm{K}} = 80$ are listed. The eigenvalue computation with a blocksize of $n_{\mathrm{BS}} = 1$ and $n_{\mathrm{B}} = 80$ of blocks needs $n_{\mathrm{sol}} = 80$ solutions to converge within $t = 33$ s. Increasing the blocksize to $n_{\mathrm{BS}} = 2$ with $n_{\mathrm{B}} = 40$ blocks enforces one restart of the iteration which leads to $n_{\mathrm{sol}} = 120$ solutions and a runtime of $t = 42$ s. A blocksize of $n_{\mathrm{BS}} = 4$ leads to $n_{\mathrm{B}} = 20$ blocks and also converges with one restart of the iteration and 120 solutions. The runtime of $t = 42$ s is also roughly the same. Further increasing the blocksize leads to additional restarts, many more solutions and longer runtimes. Here, the maximum number of restarts is set to 20. Using a blocksize of $n_{\mathrm{BS}} = 20$ and $n_{\mathrm{B}} = 4$ blocks, the required convergence tolerance is not reached within this maximum number of restarts. However, it is possible to obtain a converged solution either by increasing the maximum number of iterations or by decreasing the convergence tolerance. Within the Anasazi package of the Trilinos project, a minimum number of $n_{\mathrm{B}} = 3$ blocks is necessary which is why a blocksize of $n_{\mathrm{BS}} = 80$ and $n_{\mathrm{BS}} = 40$ cannot be chosen.

Table 3.
Influence of the blocksize on the number of solutions and the runtime during the calculation of 40 eigenvalues of the FE model of the crankshaft and 50 eigenvalues of the FE model of the carriage

	crankshaft							gear		
n_{BS}	1	2	4	5	8	10	16	1	4	6
n_{B}	80	40	20	16	10	8	5	120	30	20
n_{r}	0	1	1	2	3	6	15	0	0	1
n_{sol}	80	120	120	160	200	320	560	120	120	186
t [s]	33	42	41	51	61	91	159	665	626	846

Considering the FE model of the gear, a Krylov base of size $n_{\mathrm{K}} = 120$ is used. In Table 3 three combinations of the blocksize and the number of blocks are listed. With a blocksize of $n_{\mathrm{BS}} = 1$ and $n_{\mathrm{B}} = 120$ blocks, the iteration needs $n_{\mathrm{sol}} = 120$ solutions and no restart to converge within $t = 665$ s. Increasing the blocksize to $n_{\mathrm{BS}} = 4$ leads to $n_{\mathrm{B}} = 30$ blocks and the iteration needs $n_{\mathrm{sol}} = 120$ solutions to converge without a restart within $t = 626$ s. This is a shortening of the runtime by $\Delta t = 39$ s or roughly 6 %. Using a blocksize of $n_{\mathrm{BS}} = 6$ and $n_{\mathrm{B}} = 20$ blocks, the iterations needs to be restarted once whereby the number of solutions is increased to $n_{\mathrm{sol}} = 186$ and a runtime by $t = 846$ s. This is a growth of the runtime of $\Delta t = 181$ s or roughly 27 % compared to the computation using a blocksize of $n_{\mathrm{BS}} = 1$. To summarize, within the computations shown here, an increase of the blocksize can lead to a slight reduction of the runtime as long as a restart of the iteration is prevented. If a restart has to be done, the runtime increases heavily. Because there is no possibility to determine the largest blocksize which leads not to a restart and the small potential reduction of the runtime, in Morembs++ it is recommended to use a blocksize of $n_{\mathrm{BS}} = 1$.

In Fig. 9, the influence of the number of eigenvalues on the total runtime as well as on the runtime of the orthogonalization, the numerical factorization and the forward and backward substitution process is shown. Independent of the number of eigenvalues, the numerical factorization has to be done only once. Here, the default size $n_{\mathrm{K}} = 3\,n_{\mathrm{EV}}$ of the Krylov base is used which leads to $n_{\mathrm{sol}} = 3\,n_{\mathrm{EV}}$ assuming the iteration doesn't need to be restarted. This results to an average gradient of the runtime of the forward and backward substitution process of 2.6 s.

During each iteration, one vector is calculated which has to be orthogonalized to all previously calculated vectors. Because of that, there is a quadratic dependency between the runtime of the orthogonalization and the number of eigenvalues.

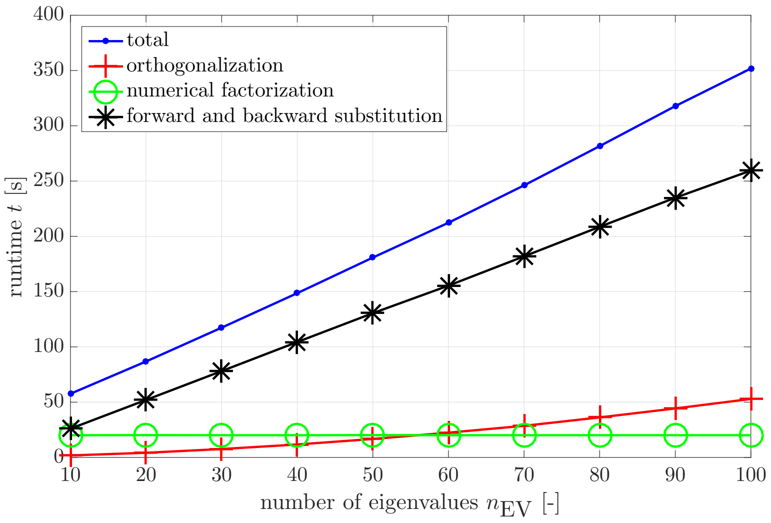

Fig. 9. Dependency between the number of eigenvalues and the runtime using the FE model of the crank

The most time consuming part of the eigenvalue computation is the forward and backward substitution process. Therefore, the total runtime behaviour is dominated by the linear behaviour of the runtime of the forward and backward substitution process and the quadratic behaviour of the runtime of the orthogonalization has only a minor influence.

The calculation of the constraint modes within the CMS based method can be done iteratively using the preconditioned CG solver [23] because the coefficient matrix is symmetric and positive definite. Here, a multigrid preconditioner from the MueLu package [24] and the CG solver from the Belos package of the Trilinos project are used. Due the positive definiteness of the symmetric matrix $\boldsymbol{K}_{\mathrm{ii}}$, the smoothed aggregation algorithm can be chosen as an optimal multigrid algorithm [25]. On the coarsest level the direct solver implemented in the MUMPS library is used. As a smoother the symmetric Gauss-Seidel relaxation method is applied with a damping factor of 0.6 and 3 sweeps. The calculation of the constraint modes using the FE model of the crankshaft leads to the solution of the system of linear equations

$$\boldsymbol{K}_{\mathrm{ii}}\boldsymbol{\Psi}_{\mathrm{ib}} = \boldsymbol{K}_{\mathrm{ib}} \tag{16}$$

with $\boldsymbol{K}_{\mathrm{ii}} \in \mathbb{R}^{444\,813\times444\,813}$ and $\boldsymbol{\Psi}_{\mathrm{ib}}, \boldsymbol{K}_{\mathrm{ib}} \in \mathbb{R}^{444\,813\times24}$. In Fig. 10 the norm of the residual $||\boldsymbol{r}||$ for each column is shown on the left at each iteration. Numerical examinations have shown that a convergence tolerance of 10^{-8} has to be reached on order to obtain an accurate result.

It can be seen that some columns need only 40 iterations until the convergence tolerance is reached while others need up to 150 iterations. The runtime of

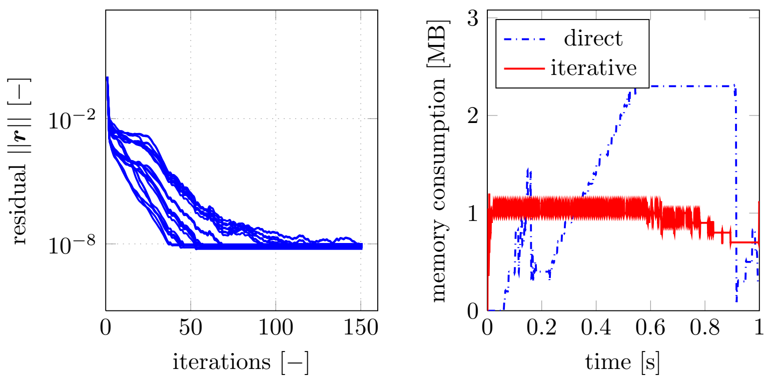

Fig. 10. The residual of each right-hand-side and the memory consumption during the iterative solution process

Morembs++ needed for this calculation of the constraint modes is 1476 s. Using the direct solver MUMPS only 56 s are needed. By applying larger FE models, like these of the gear and the carriage, the differences in the runtime are even larger. In Fig. 10 the memory consumption during the calculation of the constraint modes using the direct and iterative solver is shown on the right. The runtimes are normalized to make the comparison between the direct and iterative methods possible. The memory consumption using the direct method shows the characteristics described above. When the iterative solver is applied, only the system matrices need to be stored. There is no symbolic and numerical factorization necessary and, therefore, the memory consumption is much lower compared to when the direct solver is used. Unfortunately, the runtime using iterative methods in Morembs++ is currently too large to make the memory consumption advantage attractive.

5. Conclusion

In this work, three model reduction methods, implemented in Morembs++, have been presented as well as the process chain of Morembs++. It has been shown that, independently of the used reduction method, the most expensive step during the calculation of the projection matrix is the solution of a sparse system of linear equations. Due to the properties of the system matrices, in general only direct methods can be used in Morembs++ currently. It has been shown that the MPI parallelization is currently the only way to satisfy the huge demand memory optionally with the out-of-core functionality. Thereby, multiple computers connected in a LAN as well as in a cluster or in a supercomputer can be used. Besides that, the MPI parallelization also impacts the runtime of Morembs++. In a LAN the numerical factorization and the forward and backward substitution are the most time consuming parts and the runtime can be reduced effectively due to the MPI parallelization. On the supercomputer Hazelhen at the HLRS, it has been shown that the runtime of the numerical factorization and the forward and backward substitution can be reduced heavily by using up to 96 MPI processes. At this point, the runtime is dominated by the forward and backward substitution process. Using more than 96 MPI processes even increases the runtime of the forward and backward substitution.

Also, the size of the Krylov base as well as the blocksize within the BKS eigensolver have been examined. Coming from the default settings within the Anasazi package of the Trilinos project, the size of the Krylov base can be reduced slightly when many eigenvalues are computed. However, in Morembs++ a blocksize $n_{\mathrm{BS}} > 1$ can reduce the runtime only slightly while increasing the risk of a restart of the iteration which leads to longer runtime. Because of that, in Morembs++ the choice of a blocksize of $n_{\mathrm{BS}} = 1$ is recommended. It has also been shown that the use of iterative methods within the CMS based method is possible. Here, a preconditioned CG method was applied using a multigrid preconditioner. Thereby, the memory consumption can be decreased but the runtime is increased heavily which is why the iterative methods are currently not recommended to be used in Morembs++.

Manuscript received by Editorial Board, June 10, 2016;
final version, September 12, 2016.

References

[1] A. Antoulas. *Approximation of Large-Scale Dynamical Systems*. SIAM, Philadelphia, 2005.

[2] W. Schiehlen and P. Eberhard. *Applied Dynamics*. Springer, Heidelberg, 1 edition, 2014.

[3] A.A. Shabana. *Dynamics of Multibody Systems*. Cambridge University Press, New York, 4 edition, 2013.

[4] P. Holzwarth, M. Baumann, T. Volzer, I. Iroz, P. Bestle, J. Fehr, and P. Eberhard. Software Morembs. University of Stuttgart, Institute of Engineering and Computational Mechanics, Stuttgart, Germany, 2016. Last accessed April, 24, 2016.

[5] M. Fischer and P. Eberhard. Simulation of moving loads in elastic multibody systems with parametric model reduction techniques. *Archive of Mechanical Engineering*, 61(2):209–226, 2014.

[6] M.A. Heroux, R.A. Bartless, V.E. Howle, R.J. Hoekstra, J.J. Hu, T.G. Kolda, R.B. Lehoucq, K.R. Long, R.P. Pawlowski, E.T. Phipps, A.G. Salinger, H.K. Thornquist, R.S. Tuminaro, J.M. Willenbring, A. Williams, and K.S. Stanley. An overview of the Trilinos Project. *ACM Transactions on Mathematical Software*, 31(3):397–423, 2005.

[7] Y. Zhou and Y. Saad. Block Krylov–Schur method for large symmetric eigenvalue problems. *Numerical Algorithms*, 47(4):341–359, 2008.

[8] P. Gosselet and C. Rey. Non-overlapping domain decomposition methods in structural mechanics. *Archives of Computational Methods in Engineering*, 13(4):515–572, 2006.

[9] A. Toselli and O.B. Widlund. *Domain Decomposition Methods: Algorithms and Theory*. Springer, Heidelberg, 2005.

[10] J. Mandel. Balancing domain decomposition. *Communications in Numerical Methods in Engineering*, 9(3):233–241, 1993.

[11] C. Farhat and F.-X. Roux. A method of finite element tearing and interconnecting and its parallel solution algorithm. *International Journal for Numerical Methods in Engineering*, 32(6):1205–1227, 1991.

[12] A. Prokopenko, C.M. Siefert, J.J. Hu, M. Hoemmen, and A. Klinvex. Ifpack2 Users Guide 1.0. Technical Report SAND2016-5338, Sandia National Labs, 2016.

[13] *PERMAS, User's Reference Manual, PERMAS Version 11.00.445*. INTES Publication No. 450. INTES GmbH, Stuttgart, 2006.

[14] M. Lehner and P. Eberhard. On the use of moment-matching to build reduced order models in flexible multibody dynamics. *Multibody System Dynamics*, 16(2):191–211, 2006.

[15] P. Holzwarth and P. Eberhard. SVD-based improvements for component mode synthesis in elastic multibody systems. *European Journal of Mechanics – A/Solids*, 49:408–418, 2015.

[16] Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. SIAM, Philadelphia, 2 edition, 2011.

[17] R. Craig. Coupling of substructures for dynamic analyses: An overview. In *Proceedings of the AIAA Dynamics Specialists Conference*, Atlanta, April 5, 2000. Paper-ID 2000-1573.

[18] HDF Group: Hierarchical Data Format 5. http://www.hdfgroup.org/hdf5/. Last accessed April, 24, 2016.

[19] MUMPS: A MUltifrontal Massively Parallel Sparse Direct Solver. http://graal.ens-lyon.fr/MUMPS, 2016. Last accessed April 24, 2016.

[20] X.S. Li and J.W. Demmel. SuperLU_DIST: A scalable distributed-memory sparse direct solver for unsymmetric linear systems. *ACM Transactions on Mathematical Software*, 29(2):110–140, 2003.

[21] O. Schenk and K. Gärtner. Solving unsymmetric sparse systems of linear equations with PARDISO. *Future Generation Computer Systems*, 20(3):475–487, 2004.

[22] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[23] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, 2 edition, 2003.

[24] J.J. Hu, A. Prokopenko, C.M. Siefert, R.S. Tuminaro, and T.A. Wiesner. MueLu Multigrid Framework. http://trilinos.org/packages/muelu, 2014. Last accessed June, 8, 2016.

[25] A. Prokopenko, J.J. Hu, T.A. Wiesner, C.M. Siefert, and R.S. Tuminaro. MueLu Users Guide 1.0. Technical Report SAND2014-18874, Sandia National Labs, 2014.

Redukcja rzędu modelu w układach elementów skończonych wielkiej skali, w środowisku równoległym z intefejsem (MPI), w zastosowaniu do symulacji układów wieloczłonowych

S t r e s z c z e n i e

W ostatnich latach w symulacji układów wieloczłonowych coraz ważniejsze staje się uwzględnianie odkształcalności członów. By w symulacji układu wieloczłonowego można było wykorzystać człony odkształcalne, modelowane metodą elementów skończonych, rozmiar układu równań różniczkowych zwyczajnych musi być zredukowany drogą projekcji. W tym celu w prezentowanej pracy zastosowano metodę redukcji modalnej, metodę opartą na syntezie składowych postaciowych (CMS) oraz metodę dopasowania momentów. Wobec wciąż rosnącego rozmiaru układów niezredukowanych, obliczanie macierzy projekcji prowadzi do wielkiego zapotrzebowania na moce obliczeniowe i nie może być wykonane na zwykłych, szeregowych komputerach. W pracy zaprezentowano oprogramowanie do redukcji modelu Morembs++, w którym wykorzystuje się obliczenia równoległe z interfejsem transmisji wiadomości (MPI), co zaspokaja zapotrzebowanie na pamięć i zmniejsza czas wykonania niezbędnych obliczeń. Ponadto działanie blokowego solvera wartości własnych Kryłowa-Schura, zaimplementowanego w pakiecie oprogramowania Anasazi z projektu Trilinos, zostało przeanalizowane pod kątem wyboru rozmiaru bazy Kryłowa, rozmiaru bloku i liczby bloków. Rozważono także użycie solvera iteracyjnego w ramach metody opartej na syntezie składowych postaciowych (CMS).