Mesh Decomposition for Efficient Parallel Computing of Electrical Machines by Means of FEM Accounting for Motion

Stefan Böhmer[1], Enno Lange[1], Martin Hafner[1], Tim Cramer[2], Christian Bischof[2], and Kay Hameyer[1]

[1]Institute of Electrical Machines, RWTH Aachen University, D-52062 Aachen, Germany
[2]Center for Computing and Communication, RWTH Aachen University, D-52074 Aachen, Germany

The relative motion between stator and rotor in electrical machines requires a flexible representation in 2-D and 3-D finite element (FE) models. Numerous approaches to incorporate the relative motion are available. Due to increasing problem size and accuracy, parallel computing becomes more desirable. The parallel simulation of sufficiently large problems often involves domain decomposition algorithms, especially if distributed memory systems are used for the parallelisation. Accounting for motion usually requires explicit domain decomposition at each simulation step. This paper proposes an approach avoiding the computationally expensive domain decomposition at each step by properly choosing an initial decomposition, which can be applied to all steps throughout the simulation.

Index Terms—**Multithreading, nonconforming mesh, parallel algorithms, rigid motion.**

I. Introduction

NUMERICAL simulation of electrical machines requires a flexible implementation of the rotor motion in 2-D and 3-D models. This is especially important for parallelised code as applied in this paper. Due to the development in the microprocessor market, increase of computational power is not driven by increase of processing frequency anymore but mainly by parallelisation [1], [2]. Recent microprocessor architectures are equipped with multiple cores on every processor and additionally with multiple processors per system. Furthermore, it is possible to use computing clusters as distributed memory systems for parallelisation. Distributed memory systems require explicit domain decomposition and also multiple processor architectures can benefit from it. In this paper the parallelisation is implemented in the institute's in-house FE-package iMOOSE [www.iem.rwth-aachen.de] using a hybrid parallelisation paradigm based on OpenMP [3] and the Message Passing Interface (MPI) [4]. A recent publication of the authors regarding the hybrid parallelisation can be found in [5]. An alternative parallelisation relying on domain decomposition and particularly designed for vector processors is presented in [6].

Several approaches to simulate the motion of electric machines in FE models have been developed. Among them is the moving band technique [7], where an annulus shaped band of elements is re-generated after each rotor displacement. Since this approach is only feasible for 2-D problems, the lock-step approach is often applied in 3-D [8]. The lock-step method is a method with a regular discretization of the rotor and the stator surfaces, so that the motion can be described by associating the nodes of the rotor surface in different ways with those of the stator surface without disturbing any element in the airgap. This method requires a fixed rotational step size. Lagrange-multiplier approaches seek to overcome the disadvantages being applicable to 2-D and 3-D, static and transient problems [9].

Almost all approaches accounting for motion have in common, that degrees of freedom (DoFs) on one side are expressed as a linear combination of DoFs on the opposite side. The modified mesh decomposition presented in this paper is compatible with all motion algorithms within this category.

Manuscript received July 07, 2011; revised October 20, 2011; accepted November 04, 2011. Date of current version January 25, 2012. Corresponding author: S. Böhmer (e-mail: stefan.boehmer@iem.rwth-aachen.de)

Digital Object Identifier 10.1109/TMAG.2011.2176472

II. Adjusting Parallelization for Motion

The parallelisation implemented within the iMOOSE library is based on domain decomposition of the FE mesh, so that every involved process is exclusively working on a particular sub-mesh. The decomposition of the complete domain Ω into s subdomains Ω_i is given by:

$$\bigcup_{i=0}^{s} \Omega_i = \Omega \quad \text{with} \quad \Omega_i \cap \Omega_j = \emptyset \quad \text{for} \quad i \neq j. \tag{1}$$

The decomposition is done by a graph partitioning algorithm, e.g., by multilevel methods [10]. Therefore, the dual-graph $G = (V, E, W_v)$ corresponding to the mesh is constructed. V describes the vertex set and $E \subseteq V \times V$ the edges. W_v holds the vertex weights. Every vertex in the dual-graph represents an element of the mesh. Two vertices are adjacent if and only if the corresponding elements share a common edge in 2-D and a common face in 3-D. Additionally every vertex is weighted according to the number of DoFs, which is considered by the decomposition algorithm. After the generation of the dual-graph, the graph is partitioned by an arbitrary graph partitioning algorithm to determine the different sub-meshes. The objectives of the mesh decomposition correspond to the objectives of the partitioning algorithm in the following way: On the one hand the cut of the dual-graph resulting from the partitioning should contain as few edges as possible. Fewer edges in the cut result in fewer common DoFs in the mesh decomposition which are a quantity for the required communication between the involved processes. On the other hand the partitioning should be well balanced, which means that the sum of the vertex weights is nearly equal in all partitions. Related to the mesh decomposition this results in a equal number of DoFs in every sub-mesh and therefore in an equal amount of work for every participating process. This is a strong requirement for an efficient parallelisation because otherwise some processes would idle while others are still computing which results in a bad speedup ratio.

If motion is considered for computation, the mesh topology changes every time step at the interface between stator and rotor.

The initial mesh decomposition is only valid at the first time step and cannot be used for the other steps without modifications. One possible solution is to adjust the decomposition at every step, which results in additional communication by data traffic among all participating processes. To avoid this overhead and minimize process interdependency the mesh decomposition is modified in the proposed approach to fit any position of the motion.

A similar approach was presented in [11]. In the cited work the sub-meshes are adjusted after the mesh decomposition has been finished by assigning all elements corresponding to the moving interface to the first sub-mesh. This results in a sub-mesh which consists of two parts. On the one hand the sub-mesh computed by the mesh decomposition and on the other hand a cylinder with a thickness of two element layers representing the moving interface. In 2-D this mesh is reduced to an annulus formed band. This disturbed first sub-mesh leads to a high increase in the number of iterations required by the conjugate gradient algorithm. This increase is much higher than the general increase observed at mesh decomposition and therefore must result from this specific mesh decomposition. To overcome this problem, the proposed method in this paper tackles the problem of handling motion already during the mesh decomposition.

III. Modified Domain Decomposition

The presented modified domain decomposition for handling the motion during the parallel execution is based on the idea, that all elements corresponding to the moving interface are assigned to the same sub-mesh. In order to do so, a modification to the standard domain decomposition is required restricting the locality of the motion interface to one sub-mesh and thus, significantly reducing the communication.

Let Ω^m and Ω^s be the stator and rotor domain of an electric machine and $\mathcal{T}^m$ and $\mathcal{T}^s$ their triangulations. The airgap elements are assigned arbitrarily to stator and rotor domain. Let Γ^m and Γ^s be the interface between the stator and rotor domain and $\mathcal{T}^{\Gamma m}$ and $\mathcal{T}^{\Gamma s}$ be the elements having at least one node on the interface. Initially, a dual-graph of the discretization is constructed. Fig. 1(a) shows a zoom to the discretized air gap of a 2-D FE-model. All shown elements belong to the airgap and have been assigned to stator or rotor mesh. The corresponding dual-graph is shown in Fig. 1(b). To point out the elements responsible for the motion, all elements located at the sliding interface are highlighted by grey colour. In the next step, all vertices corresponding to elements from $\mathcal{T}^{\Gamma m} \cup \mathcal{T}^{\Gamma s}$, we call this set V^I, are merged to one supervertex v^m, cf. Fig. 1(c). The weight of v^m has to be set according to the sum of the weights of the merged vertices in order to avoid an unbalanced partitioning. Thus, a modified graph $G' = (V', E', W'_v)$ is constructed:

$$V' = V \backslash V^I \cup \{v^m\} \tag{2}$$

$$E' = \left\{\{v_j, v_k\} \,|\, \{v_j, v_k\} \in E \wedge v_j \notin V^I \wedge v_k \notin V^I\right\} \cup \left\{\{v^m, v_j\} \,|\, \{v_i, v_j\} \in E \wedge v_i \in V^I\right\} \tag{3}$$

$$W'_v = \{w_i \in W_v \,|\, v_i \in V'\} \cup \{w^m\} \tag{4}$$

$$w^m = \sum_{i, v_i \in V^I} w_i. \tag{5}$$

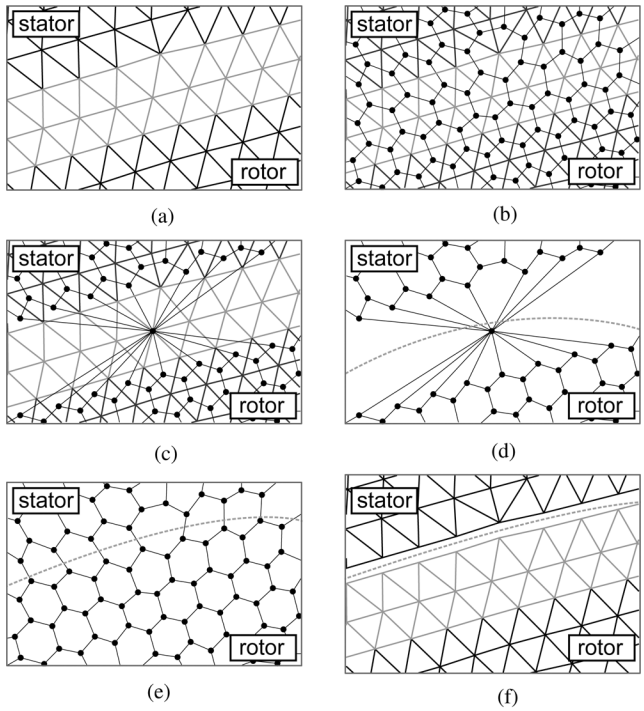

Fig. 1. Mesh decomposition taking motion into account during the partitioning of the dual-graph. (a) FE mesh, (b) FE mesh and dual-graph, (c) Merging, (d) Partitioning, (e) Expansion, (f) Decomposition.

The graph G' is partitioned by e.g., multilevel methods, which are quite fast and deliver high quality partitions [10]. The resulting cut for the given example graph is shown in Fig. 1(d). Afterwards all vertices in V^I are expanded by assigning the vertices to the same partition in which the supervertex is located, cmp. Fig. 1(e). Finally, the different sub-meshes are determined from the partitioning by the unique mapping between vertices in the dual-graph and mesh elements which has been constructed in the first step. Thereby, all elements which are located at the moving interface Γ^m and Γ^s are assigned to one single sub-mesh. The resulting mesh decomposition for the proposed example is shown in Fig. 1(f). It can be seen, that all grey elements located at the interface are assigned to the lower sub-mesh.

Only one process has to handle the motion between stator and rotor, thus no additional communication is required. The maximum reasonable number of sub-meshes is given by:

$$n_{\text{limit}} \leq \frac{|\mathcal{T}^m \cup \mathcal{T}^s|}{|\mathcal{T}^{\Gamma m} \cup \mathcal{T}^{\Gamma s}|}. \tag{6}$$

The limit n_{limit} is reached when a single sub-mesh contains all elements of the set $\mathcal{T}^{\Gamma m} \cup \mathcal{T}^{\Gamma s}$. Exceeding this limit of sub-meshes leads to unbalanced decompositions being unfavourable and limiting the efficiency of the parallelisation. This limit increases with increasing problem size. Fig. 2 shows the number of elements of a complete mesh $|\mathcal{T}^m \cup \mathcal{T}^s|$ compared to the sliding interface $|\mathcal{T}^{\Gamma m} \cup \mathcal{T}^{\Gamma s}|$ in function of different element sizes for a 2-D field problem. It can be observed, that the slope of the relative overall number of elements is quadratic with respect to the relative element size, while the slope of the number of elements on the sliding interface increases almost linearly, since

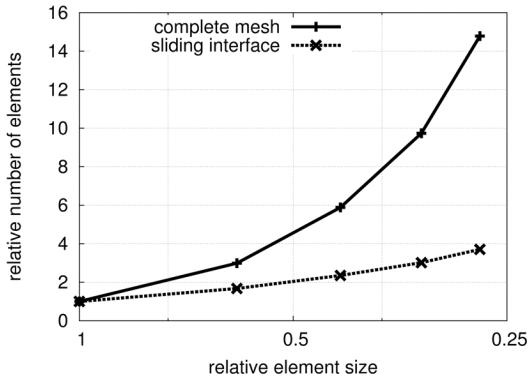

Fig. 2. Number of elements dependent on the mesh size.

this interface is a one dimensional sub-mesh. Thus, the theoretical limit n_{limit} is unlikely to be reached when simulating electrical machines using the proposed approach.

IV. Different Decomposition Strategies

To illustrate the different strategies for mesh decomposition with and without taking motion into account, both are applied to a 2-D quasi-static field problem. Fig. 3 shows the results of these strategies for a mesh decomposition into two different sub-meshes for the parallel computation with two processes. The elements located at the sliding interface between stator and rotor are drawn in black colour, whereas all other elements are coloured grey. Fig. 3(a) shows the mesh decomposition without considering the motion. The elements at the sliding interface are distributed to both sub-meshes. In contrast to this, Fig. 3(b) shows the resulting mesh decomposition, if the proposed method is applied. Using this strategy, all elements at the sliding interface of the set $\mathcal{T}^{\Gamma m} \cup \mathcal{T}^{\Gamma s}$ are assigned to one sub-mesh. For this mesh, which contains in total 10454 elements from which 414 elements are located at the sliding interface, the maximum reasonable number of sub-meshes according to (6) is 25. As a consequence parallelisation with two processes and sub-meshes is well below the possible limit for this specific FE mesh and the speedup is not influenced by the adjusted mesh decomposition. If this problem is parallelised by more than 25 processes, the speedup can be affected negatively by the mesh decomposition. However, at this low number of DoFs such a high degree of parallelisation is not reasonable in the considered implementation due to the high communication overhead [5].

V. Computational Cost

In this section the computational cost of the mesh decomposition and of the FE simulation are analysed separately, which correlate with the overall simulation time. Using this analysis, it is possible to compare the computational cost of both mesh decomposition strategies. As described in the previous sections, the mesh decomposition is done by partitioning the corresponding dual-graph. The complexity of the partitioning algorithm utilised in the proposed implementation is given by $\mathcal{O}(|V|)$ where V is the set of vertices in the dual-graph [12]. The number of vertices $|V|$ is equal to the total number of mesh elements n, as follows from the construction of the dual-graph.

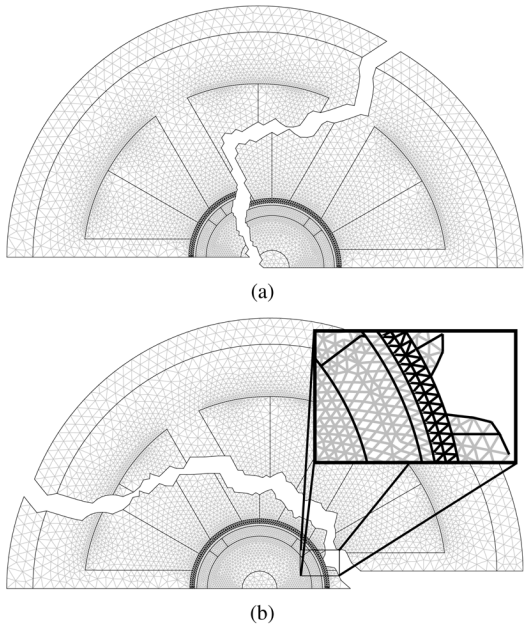

Fig. 3. Decomposition of a 2-D FE mesh into two sub-meshes. (a) Without considering motion, (b) Considering motion.

So the complexity of the mesh decomposition can be expressed by:

$$\mathcal{T}_{\mathrm{decomp}}(n) \in \mathcal{O}(n). \tag{7}$$

The FE computation consists of different parts. The complexity of some parts is given by $\mathcal{O}(n)$ like file I/O for reading the FE mesh or assembling of the element matrices. One major fraction is the solving process of the system of equations. The iMOOSE software package employs iterative methods for solving, in particular the preconditioned conjugate gradient algorithm. According to [13] the computational cost for this algorithm is $\mathcal{O}(n^{1.25})$ at 2-D problems and slightly lower at 3-D ones. This results in the following overall complexity of the FE computation:

$$\mathcal{T}_{\mathrm{FE}}(n) \in \mathcal{O}(n^{1.25}). \tag{8}$$

Comparing (7) to (8) it can be seen, that an increase in the number of elements results in a higher increase of the computation time required by the FE simulation in comparison to the mesh decomposition time. Thereby, the influence of the mesh decomposition on the total runtime decreases at increasing mesh size.

This result can be transferred to the computation time of both decomposition strategies—the proposed single mesh decomposition for all steps on the one hand and a mesh decomposition at every step on the other hand. The proposed method offers an advantage in computation time, because mesh decomposition is only executed once for all steps. This advantage decreases at increasing mesh size, because the ratio of the mesh decomposition gets smaller.

To evaluate this theoretical consideration, a 2-D field problem is investigated and the required computation times are measured for different mesh discretisations. The chosen field problem is

TABLE I
INFLUENCE OF THE NUMBER OF ELEMENTS ON THE RATIO BETWEEN MESH DECOMPOSITION AND COMPUTATION TIME

mesh	# elements	FEM	decomposition	ratio
1	22623	11.24 s	1.64 s	14.6 %
2	43325	24.37 s	2.57 s	10.6 %
3	146764	96.56 s	6.84 s	7.1 %

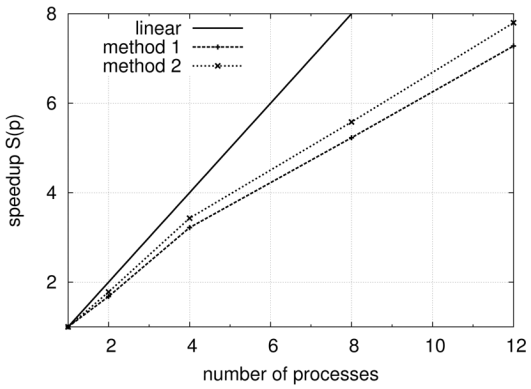

Fig. 4. Speedup of the proposed method (2) with one mesh decomposition for all steps and new mesh decomposition at every time step (1).

TABLE II
COMPUTATION TIMES OF THE PROPOSED METHOD (2) WITH ONE MESH DECOMPOSITION FOR ALL STEPS AND NEW MESH DECOMPOSITION AT EVERY TIME STEP (1)

# processes	1	2	4	8	12
method 1	8065.1 s	4794.2 s	2508.4 s	1541.3 s	1107.7 s
method 2	8065.1 s	4533.1 s	2349.8 s	1445.7 s	1034.6 s
benefit	-	5.8 %	6.7 %	6.6 %	7.1 %

the same as shown in Fig. 3. The results of the runtime measurements are presented in Table I. All measurements are executed without parallelisation utilizing only one process. Additional to the runtimes also the ratio between mesh decomposition time and simulation time is listed. This ratio decreases with increasing number of elements from 14.6% at 22 623 elements to 7.1% at 146 764 elements. If the mesh decomposition and the FE simulation are both parallelised, this ratio keeps roughly equal.

VI. APPLICATION 3-D

In addition to the 2-D field problem in Section IV, the proposed method has been applied to a 3-D quasi-static field problem of a Permanent Magnet Synchronous Machine (PMSM), which consists of about 700 000 elements. The lock-step method is used to take motion into account. To evaluate the advantages of the described method in terms of the speedup ratio, it is compared to the approach, where a new mesh decomposition is generated at every time step by repartitioning. Table II shows the measured runtimes for a computation consisting of 10 time steps. The method requiring a new mesh decomposition at every time step is denoted by method 1 and the proposed method is denoted by method 2. The computation has been parallelised with up to 12 processes assigning one process to every core. In the sequential case using only one process, the times of both methods are equal, because there is no mesh decomposition. The gain of the proposed method ranges between 5% and 7%. This benefit can also be observed at the speedup ratio shown in Fig. 4.

VII. CONCLUSION

This paper proposes a modification to the standard domain decomposition in parallel computing of electrical machines. By introducing a weighted, virtual supervertex in the dual-graph representing the elements on the sliding air gap interface, single domain decomposition yields proper sub-meshes for all subsequent simulation steps. The described method requires only a single decomposition of the FE mesh and results in a decreasing simulation time, which is the main objective in parallelisation. Hereby, an efficient parallel computation of electrical machines accounting for motion on distributed memory systems can be implemented.

Furthermore, the limit regarding the scalability of the parallelisation exploiting the given mesh decomposition is analysed. It is shown, that the theoretic limit is not relevant in practice and does not affect the possible scalability. The application of the proposed method to a 3-D field problem shows the advantages with respect to the parallelisation, in particular smaller computation time and higher speedup ratio.

REFERENCES

[1] M. J. Flynn and P. Hung, "Microprocessor design issues: Thoughts on the road ahead," *IEEE Micro*, vol. 25, pp. 16–31, May 2005.
[2] C. H. Bischof, "Parallel computers everywhere," in *16th Int. Conf. Computation of Electromagnetic Fields, Compumag 2007*, Aachen, Jun. 2007.
[3] OpenMP Application Program Interface. 3rd ed. OpenMP Architecture Review Board, [Online]. Available: http://www.openmp.org/mpdocuments/spec30.pdf, May 2008
[4] MPI-2 Extensions to the Message Passing Interface. 2nd ed. Message Passing Interface Forum, [Online]. Available: http://www.mpi-forum.org, 1997
[5] S. Böhmer, T. Cramer, M. Hafner, E. Lange, and K. Hameyer, "Numerical simulation of electrical machines by means of a hybrid parallelisation using MPI and OpenMP for FEM," in *Int. Conf. Computation in Electromagnetics, CEM 2011*, Wroclaw, Poland, Apr. 2011, pp. 180–181.
[6] T. Nakano, Y. Kawase, T. Yamaguchi, M. Nakamura, N. Nishikawa, and H. Uehara, "Parallel computing of magnetic field for rotating machines on the earth simulator," *IEEE Trans. Magn.*, vol. 46, no. 8, pp. 3273–3276, 2010.
[7] B. Davat, Z. Ren, and M. Lajoie-Mazenc, "The movement in field modeling," *IEEE Trans. Magn.*, vol. 21, no. 6, pp. 2296–2298, 1985.
[8] T. Preston, A. Reece, and P. Sangha, "Induction motor analysis by time-stepping techniques," *IEEE Trans. Magn.*, vol. 24, no. 1, pp. 471–474, 1988.
[9] E. Lange, F. Henrotte, and K. Hameyer, "A variational formulation for nonconforming sliding interfaces in finite element analysis of electric machines," *IEEE Trans. Magn.*, vol. 46, no. 8, pp. 2755–2758, 2010.
[10] G. Karypis and V. Kumar, "Multilevel algorithms for multi-constraint graph partitioning," in *Proc. 1998 ACM/IEEE Conf. Supercomputing*, Washington, DC, 1998, pp. 1–13, ser. Supercomputing'98, IEEE Computer Society.
[11] D. Walsh, C. Glasgow, and I. Dawkins, "Modeling rotating machines on parallel computers using the bsp library," *IEEE Trans. Magn.*, vol. 35, no. 3, pp. 1286–1289, May 1999.
[12] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Sci. Comput.*, vol. 20, pp. 359–392, 1998.
[13] M. T. Heath, *Scientific Computing: An Introductory Survey*, E. M. Munson, Ed., 2nd ed. New York: McGraw-Hill, 2002.