Parallel-in-time Simulation of Eddy Current Problems using Parareal

Sebastian Schöps[1,2], Innocent Niyonzima[1,2], and Markus Clemens[3]
[1]Technische Universität Darmstadt, Institut für Theorie Elektromagnetischer Felder,
Schlossgartenstrasse 8, D-64289 Darmstadt, Germany
[2]Technische Universität Darmstadt, Graduate School of Computational Engineering,
Dolivostrasse 15, D-64293 Darmstadt, Germany
[3]Bergische Universität Wuppertal, Chair of Electromagnetic Theory,
Rainer-Grünter-Strasse 21, D-42119 Wuppertal, Germany

In this contribution the usage of the Parareal method is proposed for the time-parallel solution of the eddy current problem. The method is adapted to the particular challenges of the problem that are related to the differential algebraic character due to non-conducting regions. It is shown how the necessary modification can be automatically incorporated. The paper closes with a first demonstration of a simulation of a realistic four-pole induction machine model using Parareal.

***Index Terms*—Parallel-in-time, eddy current, time stepping, finite elements**

I. Introduction

THE numerical simulation of the eddy current problem in time domain is computationally expensive due to implicit time stepping. This is particularly challenging if long time periods have to be considered as for example when the start-up of an electrical machine is simulated. In this contribution the usage of the Parareal method [1] is proposed and its application to a real world electrical engineering problem is shown. Furthermore, the method is adapted to the particular challenges of space discretized eddy currents problems related to the differential algebraic character of the equation.

When disregarding displacement currents, and introducing the magnetic vector potential $\vec{A}$ as unknown, one obtains the eddy current problem in A*-formulation

$$\sigma\frac{\partial\vec{A}}{\partial t}+\nabla\times(\nu\nabla\times\vec{A})=\vec{\chi}_{\mathrm{s}}\mathbf{i} \tag{1}$$

on the domain $\Omega\times\mathcal{I}$ and $\mathcal{I}:=(t_0,t_{\mathrm{end}}]$. The problem is well posed when supplying a gauge condition, suitable boundary conditions and an initial value $\vec{A}(\vec{r},t_0)=\vec{A}_0(\vec{r})$. The material is described by conductivity σ and nonlinear reluctivity ν; $\vec{\chi}_{\mathrm{s}}$ is the stranded-conductor winding function, which homogeneously distributes the current in the domain, i.e., $\vec{J}_{\mathrm{s}}=\vec{\chi}_{\mathrm{s}}\mathbf{i}$ where $\mathbf{i}$ is the vector of n_{s} time-dependent lumped currents.

Applying Finite Elements to the current-driven scenario (1) yields the following semi-discrete system of differential algebraic equations (DAEs)

$$\mathbf{M}_\sigma\mathbf{a}'(t)+\mathbf{K}_\nu\big(\mathbf{a}(t)\big)\mathbf{a}(t)=\mathbf{X}_{\mathrm{s}}\mathbf{i}(t) \tag{2}$$

where $\mathbf{a}(t)\in\mathbb{R}^n$ is the vector of line-integrated magnetic vector potentials, $\mathbf{M}_\sigma\in\mathbb{R}^{n\times n}$ denotes the (singular) mass matrix representing the conductivities, $\mathbf{K}_\nu(\mathbf{a})\in\mathbb{R}^{n\times n}$ is the flux-dependent curl-curl matrix and $\mathbf{X}_{\mathrm{s}}\in\mathbb{R}^{n\times n_{\mathrm{s}}}$ the discretized winding function. Movements in the computational domain are considered by the moving band approach.

The solution of (2) is straight forward if the additional equation for the voltages is not considered, i.e., it is an index-1 DAE [4]. This can be treated with standard techniques, while higher index problems are increasingly more difficult to solve.

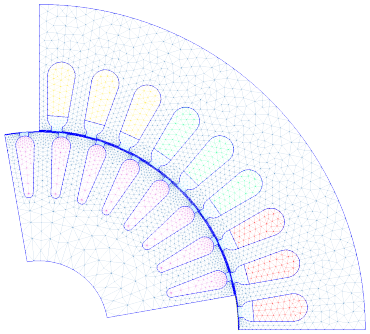

Figure 1. Mesh view of the four-pole induction machine model "im_3kw" from the GetDP library [2] as described in [3].

Let $\mathbf{M}_\sigma^+$ be the Moore-Penrose pseudo inverse of $\mathbf{M}_\sigma$ such that $\mathbf{P}_\sigma=\mathbf{M}_\sigma^+\mathbf{M}_\sigma$ and $\mathbf{Q}_\sigma=\mathbf{I}-\mathbf{P}_\sigma$ denote projectors decomposing the vector potential $\mathbf{a}_i=\mathbf{a}(t_i)$ at each time instance into its differential and algebraic components, respectively

$$\mathbf{a}_i=\mathbf{P}_\sigma\mathbf{a}_{i,\sigma}+\mathbf{Q}_\sigma\mathbf{a}_{i,0}.$$

When solving (2) for given currents $\mathbf{i}_i=\mathbf{i}(t_i)$, only initial conditions for the differential components $\mathbf{a}_{0,\sigma}$ may be prescribed. The algebraic components $\mathbf{a}_{0,0}$ must be consistently determined by solving the constraint

$$\mathbf{Q}_\sigma^\top\mathbf{K}_\nu\mathbf{Q}_\sigma\mathbf{a}_{0,0}=\mathbf{Q}_\sigma^\top\mathbf{X}_{\mathrm{s}}\mathbf{i}_0-\mathbf{Q}_\sigma^\top\mathbf{K}_\nu\mathbf{P}_\sigma\mathbf{a}_{0,\sigma}. \tag{3}$$

However, when using the implicit Euler method to solve an initial value problem with inconsistent data, i.e, $\mathbf{a}_{0,\sigma}$ and $\mathbf{a}_{0,0}$ do not fulfill (3), a projection is automatically carried out: the time stepping instruction for t_i to $t_{i+1}=t_i+\delta t$

$$\Big(\frac{1}{\delta t}\mathbf{M}_\sigma+\mathbf{K}_\nu\big(\mathbf{a}_{i+1}\big)\Big)\mathbf{a}_{i+1}=\mathbf{X}_{\mathrm{s}}\mathbf{i}_{i+1}+\frac{1}{\delta t}\mathbf{M}_\sigma\mathbf{a}_i \tag{4}$$

ignores inconsistent algebraic components after the first step due to the term $\mathbf{M}_\sigma\mathbf{a}_i=\mathbf{M}_\sigma\mathbf{P}_\sigma\mathbf{a}_i=\mathbf{M}_\sigma\mathbf{a}_{i,\sigma}$. This is generally not the case for higher index DAEs and other time-stepping schemes, see e.g. [4].

Algorithm 1: Parareal as proposed in [1].

1 initialize: $\mathbf{a}_0^{(k)} \leftarrow \mathbf{a}_0$ and $\bar{\mathbf{a}}_j^{(0)}, \tilde{\mathbf{a}}_j^{(0)} \leftarrow \mathbf{0}$ (for all j, k);
2 set counter: $k \leftarrow 1$;
3 **while** $\max_j \|\mathbf{a}_j^{(k)} - \mathbf{a}_j^{(k-1)}\| > tol$ **do**
4 **for** $j \leftarrow 1$ **to** n_{cpu} **do**
5 solve coarse problem: $\bar{\mathbf{a}}_j^{(k)} \leftarrow \mathcal{G}(T_j, T_{j-1}, \mathbf{a}_{j-1}^{(k)})$;
6 post process: $\mathbf{a}_j^{(k)} \leftarrow \bar{\mathbf{a}}_j^{(k)} + \tilde{\mathbf{a}}_j^{(k-1)} - \bar{\mathbf{a}}_j^{(k-1)}$;
7 **end**
8 **parfor** $j \leftarrow 1$ **to** n_{cpu} **do**
9 solve fine problem: $\tilde{\mathbf{a}}_j^{(k)} \leftarrow \mathcal{F}(T_j, T_{j-1}, \mathbf{a}_{j-1}^{(k)})$;
10 **end**
11 increment counter: $k \leftarrow k + 1$;
12 **end**

II. The Parareal Method

Parareal, as shown in Alg. 1, takes advantage of the parallel architecture of modern computers to speed up the time integration similar to multiple shooting methods [1]. The time interval is split into time intervals $\mathcal{I}_j := (T_{j-1}, T_j]$ with $t_0 = T_0 < T_1 < \ldots < T_{n_{\text{cpu}}} = t_{\text{end}}$ according to the number of CPUs n_{cpu} available. Two types of problems are solved in a nested loop until convergence is reached, e.g. by the implicit Euler method (4): a cheap problem defined on a coarse time and possibly spatial grid is solved sequentially (line 5, Alg. 1) to propagate missing initial conditions and high-fidelity problems are solved in parallel on the intervals $\mathcal{I}_j$ (line 9, Alg. 1).

In the algorithm, the solution operator of the cheap problem is denoted by $\mathcal{G} : \mathcal{I} \times \mathcal{I} \times \mathbb{R}^n \longrightarrow \mathbb{R}^n$ such that

$$\bar{\mathbf{a}}_j = \mathcal{G}(T_j, T_{j-1}, \bar{\mathbf{a}}_{j-1}) \tag{5}$$

which computes $\bar{\mathbf{a}}_{j+1}$ at T_{j+1} by propagating the initial value $\bar{\mathbf{a}}_j$ from T_j to T_{j+1} by coarsely discretizing (2) in time, i.e., using large δt. Similarly the solution operator of the high-fidelity problem is given by $\mathcal{F} : \mathcal{I} \times \mathcal{I} \times \mathbb{R}^n \longrightarrow \mathbb{R}^n$

$$\tilde{\mathbf{a}}_j = \mathcal{F}(T_j, T_{j-1}, \tilde{\mathbf{a}}_{j-1}) \tag{6}$$

which computes $\tilde{\mathbf{a}}_{j+1}$ by solving (2) with initial condition $\tilde{\mathbf{a}}_j$ using fine discretizations, i.e., small δt, see Alg. 1.

For the eddy current problem line 6 of Alg. 1 must be changed to reflect the differential algebraic character, i.e.

$$\mathbf{P}_\sigma \mathbf{a}_j^{(k)} = \mathbf{P}_\sigma \bar{\mathbf{a}}_j^{(k)} + \mathbf{P}_\sigma \tilde{\mathbf{a}}_j^{(k-1)} - \mathbf{P}_\sigma \bar{\mathbf{a}}_j^{(k-1)}$$

with a subsequent solve of (3) to obtain a consistent $\mathbf{Q}_\sigma \mathbf{a}_j^{(k)}$. However, when using Implicit Euler as shown above, this step is automatically taken care of. Similarly, the norm in line 3 should be adapted to only account for differential components, e.g. by considering a projection or the eddy current losses.

III. Discussion

The algorithms were implemented in Octave [5] while GetDP is used for the simulation of the four-pole induction machine model "im_3kw" with 8308 degrees of freedom [2], [3]. They are executed on an Intel Xeon cluster with 80×2.00GHz cores, i.e., $8\times$E7-8850 and 1TB DDR3 memory. As sequential time stepper the implicit Euler scheme is applied

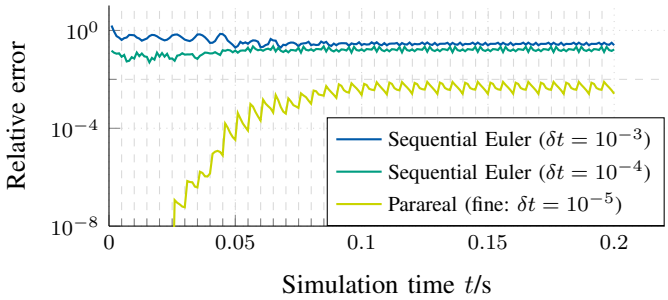

Figure 2. Relative l_2 errors during time; dashed vertical lines show the 40 intervals of size $h = 0.05$ s on which the problem is solved in parallel

for 10 electrical periods ($\mathcal{I} = (0, 0.2]$ s). The time grid of this simulation is refined from $\delta t = 10^{-3}$ s and $\delta t = 10^{-4}$ s to $\delta t = 10^{-5}$ s. The coarsest sequential simulation takes ca. 15min, while the ones with finer grids correspondingly more time, i.e., 2h and 20h, respectively.

The Parareal implementation uses OpenMP parallelized calls of GetDP. The implicit Euler method is used with time step sizes $\delta t = 10^{-5}$ s and $\delta t = 10^{-3}$ s for the fine and coarse problem, respectively. Fig. 2 shows the errors in comparison with the sequential reference simulation at $\delta t = 10^{-5}$ s. Only $n_{\text{cpu}} = 40$ of the 80 cores have been used. After $k = 4$ Parareal iterations a relative l_2 accuracy of 10^{-2} has been obtained. This corresponds to a potential speed-up of $n_{\text{cpu}}/k = 10$ with respect to the reference simulation when neglecting communication costs and the coarse grid solution. The speed-up is obtained since the effective length of the time interval was reduced by a factor of $n_{\text{cpu}} = 40$ but iterated $k = 4$ times. However, due to suboptimal implementation the actual speed-up was only 2-3. The full paper will discuss improvements by using more sophisticated solvers on the coarse grid, implementations and advantages using OpenMPI.

Acknowledgement

This work was supported by DFG grants SCHO 1562/1-1 and CL 143/11-1, the Excellence Initiative of German Federal and State Governments and the Graduate School for Computational Engineering at Technische Universität Darmstadt.

References

[1] J.-L. Lions, Y. Maday, and G. Turinici, "A parareal in time discretization of PDEs", *Comptes Rendus de l'Académie des Sciences – Series I – Mathematics*, 332, no., pp. 661–668, 2001.

[2] C. Geuzaine, "GetDP: A general finite-element solver for the de Rham complex", in *PAMM*, vol. 7, Wiley, Dec. 2007, pp. 1 010 603–1 010 604.

[3] J. Gyselinck, L. Vandevelde, and J. Melkebeek, "Multi-slice FE modeling of electrical machines with skewed slots-the skew discretization error", *IEEE Trans. Magn.*, 37, no., pp. 3233–3237, Sep. 2001.

[4] A. Bartel, S. Baumanns, and S. Schöps, "Structural analysis of electrical circuits including magnetoquasistatic devices", *APNUM*, 61, no., pp. 1257–1270, Sep. 2011.

[5] J. W. Eaton, D. Bateman, S. Hauberg, *et al.*, *The GNU Octave 4.0 Reference Manual 1/2: Free Your Numbers*. Samurai Media Limited, Oct. 2015.